ALGEBRAIC MULTIGRID FOR DISCRETE DIFFERENTIAL FORMS

BY

WILLIAM N. BELL

B.S., Georgia Institute of Technology, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Abstract

Discrete differential forms arise in scientific disciplines ranging from computational electromagnetics to computer graphics. Examples include stable discretizations of the eddy-current problem, topological methods for sensor network coverage, visualization of complex flows, surface parameterization, and the design of vector fields on meshes. In this thesis we describe efficient and scalable numerical solvers for discrete k-form problems. Our approach is based on the principles of *algebraic multigrid* (AMG) which is designed to solve large-scale linear systems with optimal, or near-optimal efficiency. Since the k-form problems to be solved are arbitrarily large, the need for scalable numerical solvers is clear.

Dedicated to my parents.

Acknowledgments

I am grateful to the faculty and students of the Scientific Computing Group in the Department of Computer Science at the University of Illinois at Urbana-Champaign for creating an atmosphere of collaboration and cohesion. In particular, I extend thanks to Luke Olson, my advisor, for his advice, insight, and dedication to our past and ongoing research efforts. Also, kudos to Stephen Bond for maintaining an open-door policy despite constant pestering from myself and others. I sincerely appreciate the friendship of my fellow Computer Science cohorts including: Bill Cochran, Andrew Colombi, Eric Cyr, Shen Dong, Jared Hoberock, and Jacob Schroder among many others. I thank Yizhou Yu and Anil Hirani for their contributions to my graduate education.

I was fortunate to meet Peter Mucha in my final year of undergraduate studies at Georgia Tech. Our collaboration on granular material simulation piqued my interest in scientific research and ultimately led to my first publication. Without question, Peter is the individual most responsible for my decision to pursue a graduate degree.

A special mention goes to Bruce Conover at Fort Myers High School. Interspersed with wit and anecdotes about Mesoamerican cultures, mathematics with "Mr. Conover", as he was known to his students, was the highlight of my secondary education.

I thank my parents for their unwavering support and encouragement throughout my twenty-two year academic career. In hindsight, I am especially grateful for the time my mother pleaded with the high-school administration for my admission to the IB program. Agreeing that I was a "late bloomer", she made the case that my poor grades didn't represent my full potential. I like to think she was right. I have little doubt that my father's enthusiasm for technology, which provided me with an early exposure to programming and computers, shaped my career choices. I am indebted to William and Valerie Bell for their support and devotion.

Lastly, I am indebted to my wife for her indomitable spirit and cheerful demeanor during the last few years. A counterweight to my (equally indomitable) cynicism, Elizabeth has been an indispensable companion through disappointments and periods of anxiety. I look forward to sharing the rest of life's adventure with you.

Table of Contents

List of Tables

List of Algorithms

List of Abbreviations

AMG	Algebraic Multigrid
SA	Smoothed Aggregation
WPD	Work Per Digit of Accuracy
OC	Operator Complexity
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation

List of Symbols

∂	Boundary operator
$\mathcal{H}$	Space of Harmonic forms
$\boldsymbol{d}$	Exterior derivative
$\mathbb{D}$	Discrete exterior derivative
$\boldsymbol{\delta}$	Codifferential (adjoint of $\boldsymbol{d}$)
$*$	Hodge star
$\mathbb{M}$	Discrete Hodge star (k-form innerproduct)
$\boldsymbol{\Delta}$	Laplacian operator
$\boldsymbol{\nabla}$	Gradient operator
$\boldsymbol{\nabla}\cdot$	Divergence operator
$\boldsymbol{\nabla}\times$	Curl operator
$\mathcal{I}$	Interpolation map from discrete cochains to k-forms
$\mathcal{R}$	de Rham map from smooth k-forms to discrete cochains
$\mathcal{P}$	Prolongation operator
$\mathcal{S}$	Prolongation smoother
$\mathcal{R}$	Range space
img	Image
span	Span of a basis
diag	Diagonal of a matrix
nnz	Number of nonzeros in a sparse matrix

Chapter 1

Introduction

Discrete differential forms arise in scientific disciplines ranging from computational electromagnetics to computer graphics. Examples include stable discretizations of the eddy-current problem [46, 12, 4], topological methods for sensor network coverage [16], visualization of complex flows [36, 41], surface parameterization [25], and the design of vector fields on meshes [21]. In this thesis we describe efficient and scalable numerical solvers for discrete k-form problems. Our approach is based on *algebraic multigrid* (AMG) [42] principles, which are designed to solve large-scale linear systems with optimal, or near-optimal efficiency. Since the k-form problems to be solved are arbitrarily large, the need for scalable numerical solvers is clear.

1.1 Overview of Ideas

This thesis develops efficient multigrid solvers for discrete differential forms. The main contributions of the thesis are two multigrid solvers, the *chain complex method* (cSA) and the *k-form basis method* (kSA). Additionally, we develop Lloyd aggregation, a novel aggregation algorithm appropriate for k-form problems. Furthermore, we advance the computation of discrete Hodge decompositions using the proposed solvers, with application to a broad class of mimetic discretizations [10] and purely topological problems. Finally, we consider the sensor network coverage problem and introduce a solver that significantly improves upon previous methods.

The chain complex method, which coarsens discrete forms in a manner that preserves their differential structure, is a natural extension of the work of Reitzinger and Schöberl [37] from 1-forms to general k-forms. Further generalizations of the basic method to improved interpolation operators [29] and the cochain complex are facilitated by examining the coarsening procedure in an algebraic setting.

Our second contribution, the k-form basis method (kSA), is motivated by the observation that performance of the cSA method degrades when moving from structured to unstructured meshes. Inspired by the work of Bochev et al. [8], kSA uses a set of k-form basis functions to guide the construction of interpolation operators. The primary distinction between the cSA and kSA

methods is that the former maintains the differential structure throughout the multigrid hierarchy while the latter ensures that certain k-forms are accurately represented at all levels.

Since the matrix structure of problems arising in k-form discretizations differs from that of conventional finite elements, a novel aggregation algorithm based on Lloyd's method is introduced. Lloyd aggregation facilitates the efficient construction of the kSA multigrid hierarchy. Furthermore, Lloyd aggregation supports a time-memory tradeoff between the cost of the multigrid hierarchy and the efficiency of the resulting multigrid cycle.

1.2 Organization

Chapter 2 introduces the general principles of algebraic multigrid and explains standard multigrid terminology. The construction of algebraic multigrid solvers based on smoothed aggregation is described in detail. An overview of discrete differential forms is presented in Chapter 3. Here, the relationship between continuous and discrete k-forms and their associated operators is explored and the scope of problems to be considered is established.

Using the discrete de Rham complex, Chapter 4 develops the chain complex method (cSA) for solving k-form Laplacians. The cSA method creates a hierarchy of progressively coarser levels that preserve the structure of the de Rham complex. In particular, the discrete derivative operators within each level of the hierarchy form a chain complex. Furthermore, interpolation operators that act between levels of the hierarchy and commute with the coarse- and fine-level discrete derivatives are constructed. The interpolation operators serve as the tentative prolongators within the smoothed aggregation methodology. Finally, we demonstrate how cSA is used to efficiently solve linear systems with the discrete k-form Laplacian operators. This chapter includes the results of a previously published work [6].

Chapter 5 discusses the primary deficiency of the chain complex method. The k-form basis method (kSA) addresses this problem by ensuring that a k-form basis is accurately represented by the multigrid prolongators. Coarse basis functions of the cSA and kSA methods are compared and a comparative numerical study is conducted. Performance of the kSA method is shown to surpass that of the cSA method in the context of problems discretized on unstructured meshes. An adaptive kSA method and its application to combinatorial Laplacians are also considered.

A novel aggregation method, Lloyd aggregation, is the subject of Chapter 6. Lloyd aggregation is motivated by a need to aggregate the degrees of freedom in k-form problems efficiently. A numerical study is conducted comparing the performance of Lloyd aggregation to the standard algorithm. When applied to k-form problems the proposed method exhibits superior performance. Furthermore, Lloyd aggregation supports a time-memory tradeoff between work per

digit of accuracy and operator complexity. Aggregation in the kSA method utilizes the proposed algorithm.

The discrete Hodge decomposition is introduced in Chapter 7. Application of the discrete Hodge decomposition to fluid simulation, visualization, and topology are discussed. A distinction is drawn between the non-physical but practically important "special" case using a trivial innerproduct and "general" case which admits an arbitrary innerproduct. The cohomology basis, or basis for the harmonic forms on a manifold, is shown to play a central role in the Hodge decomposition. Equations arising in the discrete Hodge decomposition are solved with the cSA and kSA methods.

Chapter 8 discusses the sensor network coverage problem and introduces a coordinate-free approach to its solution. The coordinate-free approach reduces the geometric coverage problem to a problem of computational topology. Here, the Rips complex, which is constructed using only communication between nearby sensors, is the topological entity of interest. Determining sufficient conditions on network coverage reduces to computing nullvectors of combinatorial Laplacians associated to the Rips complex. The proposed multigrid solver is shown to offer profound improvements over previously applied numerical methods. Finally, the main contributions of the thesis are reiterated in Chapter 9.

Chapter 2

Algebraic Multigrid

Large-scale problems in the computational sciences necessitate efficient numerical solvers. *Multigrid methods* are designed to solve systems with optimal time (and space) complexity. Specifically, the number of operations needed to solve a problem scales linearly with the size of the problem. Although the multigrid methodology may be applied to linear and nonlinear equations[1], in this thesis we restrict our attention to the linear case $Ax = b$.

The fundamental principle of multigrid is the complementary relationship between *relaxation* and *coarse-grid correction*. When applied to a linear system, relaxation methods (e.g. Gauss-Seidel iteration) rapidly reduce high-frequency or oscillatory errors in the approximate solution. After several applications of the relaxation procedure the approximate solution consists primarily of low-frequency or slow-to-converge error components that are not effectively reduced by further relaxation. However, when these components are restricted to a coarser grid they become oscillatory and are exposed to relaxation applied to the coarser problem. The approximate solution on the coarse grid is then interpolated back to the finer grid to update the fine-grid approximate solution. This procedure is applied recursively to form a *hierarchy* of grids until the coarsest problem can be solved exactly. The processes of relaxation and coarse-grid correction comprise the *multigrid cycle*. The efficiency of the multigrid cycle depends on the effectiveness of relaxation on oscillatory error modes and the accuracy with which slow-to-converge error components are represented on coarser grids. Algorithm 2.1 describes the steps of a multigrid V-cycle.

In *geometric multigrid* [42] coarser grids are determined by a predefined geometric simplification. For instance, if a problem is discretized on a regular grid with N subdivisions in each dimension, a natural geometric coarsening has $N/2$ subdivisions in each dimension. Restriction and interpolation are then defined in a manner consistent with the coarsening of the domain. Often, the coarse level equations correspond to coarser discretizations of the initial (continuous) problem.

In contrast, *algebraic multigrid* (AMG) [38, 44] applies multigrid principles directly to the linear system of the discrete problem $Ax = b$. In place of a fixed geometric coarsening, AMG develops coarse grids[2] to accurately capture relax-

[1] e.g. the Full Approximation Scheme (FAS).

[2] Although not proper grids, the same terminology is still used in AMG.

Algorithm 2.1: `multigrid_cycle`$(A_0, \ldots, A_N, x_l, b_l)$

```
1  if l = N
2      return Solve(A_l, b_l)
3  else
4      x_l ⇐ Presmooth(A_l, x_l, b_l)
5      r_l ⇐ b_l − A_l x_l
6      b_{l+1} ⇐ Restrict(r_l)
7      x_{l+1} ⇐ 0
8      x_{l+1} ⇐ multigrid_cycle(A_0, ..., A_N, x_{l+1}, b_{l+1})
9      x_l ⇐ x_l + Interpolate(x_{l+1})
10     x_l ⇐ Postsmooth(A_l, x_l, b_l)
11     return x_l
12 end
```

Algorithm 2.2: `sa_hierarchy`(A, B)

```
1  A_0 ⇐ A
2  B_0 ⇐ B
3  for n = 0, 1, ..., N
4      Agg_n ⇐ Aggregate(A_n)
5      T_n, B_{n+1} ⇐ FitCandidates(Agg_n, B_n)
6      P_n ⇐ SmoothProlongator(A_n, T_n)
7      A_{n+1} ⇐ P_n^T A_n P_n
8  end
9  return A_0, ..., A_N, P_0, ..., P_{N−1}
```

ation's slow-to-converge modes (so-called algebraically smooth errors). AMG methods may be applied to anisotropic problems and problems discretized on unstructured meshes without making special considerations. Furthermore, one AMG method may solve discrete equations for a number of different PDEs with minimal modification. For problems amenable to geometric multigrid, a special-purpose geometric method will generally outperform an algebraic method on the basis of computational efficiency. However, in practice, the flexibility, robustness, and breadth of algebraic solvers often outweighs runtime efficiency considerations.

2.1 Smoothed Aggregation

In this section we briefly outline the standard components of AMG based on *smoothed aggregation* (SA)[44] using Algorithm 2.2 as a reference. Once established, the multigrid methods k-form problems described in Chapters 4 and 5 are presented as extensions of the smoothed aggregation approach.

In order to construct a multigrid hierarchy with SA one requires a set of K *near-nullspace candidates* $b_0, b_1, \ldots, b_{K-1}$ in addition to the matrix A of the linear system. While the number and character of the near-nullspace vectors

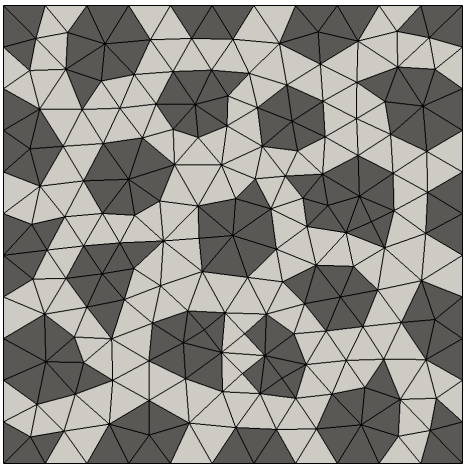

Figure 2.1: Typical aggregates on an unstructured triangle mesh.

vary from problem to problem their role is always the same: representing locally smooth functions. For instance, when solving the Poisson problem $-\Delta u = f$ a single constant candidate vector $b_0 = [1, 1, \ldots, 1]^T$ suffices to describe locally smooth functions. In linearized elasticity, the rigid-body modes (translations and rotations) comprise the near-nullspace modes for a total of 3 vectors in two-dimensions and 6 vectors in three-dimensional elasticity. In the absence of Dirichlet boundary conditions, the candidates constitute a nullspace basis of their respective problems. By convention, the K candidate vectors are assembled into the columns of a block matrix $B = [b_0, b_1, \ldots, b_{K-1}]$. In subsequent steps, the candidate vectors will be used to ensure that interpolation accurately captures non-oscillatory error modes.

In the first step of SA, the structure of A is used to decompose the computational domain into aggregates such as those shown in Figure 2.1. In the simplest case, aggregates are formed based on the nonzero pattern in A alone. However, for many problems it is necessary to apply a *strength-of-connection* measure before aggregation. In this case, only *strongly connected* degrees of freedom are aggregated together. Intuitively, strong connections are those along which algebraically smooth error varies slowly. The development of robust strength-of-connection measures remains an active research area [39].

In Algorithm 2.2 the result of aggregation on the n-th level is denoted Agg_n. Sparse matrix Agg_n maps degrees of freedom to their corresponding aggregates. Specifically, the value of $Agg_n(i, j)$ is one if the i-th degree of freedom belongs to the j-th aggregate. Otherwise, $Agg_n(i, j)$ takes the value zero. For example,

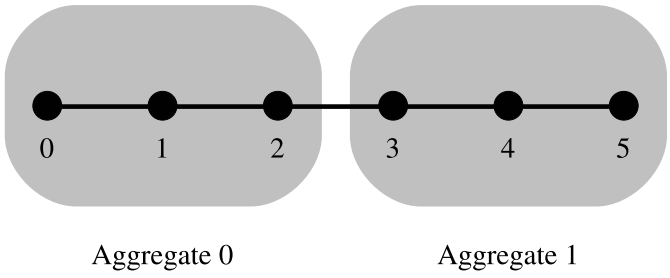

Figure 2.2: One dimensional mesh with two aggregates.

the matrix

$$Agg_0 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \tag{2.1}$$

encodes the aggregation in Figure 2.2.

The sparsity pattern of Agg_n determines the support of the coarse-level basis functions in T_n, the tentative prolongator. Additionally, the tentative prolongator T_n and coarse-level candidates B_{n+1} produced by `FitCandidates` are chosen to satisfy $T_n B_{n+1} = B_n$, i.e. the tentative prolongator exactly interpolates the fine-level candidates. When multiple candidate vectors are used, the columns of T_n are orthonormalized over each aggregate. Specifically, a QR decomposition of the K candidate vectors restricted to each aggregate is computed. Continuing the example in Figure 2.2 with fine-level candidates

$$B_0 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}, \tag{2.2}$$

the corresponding tentative prolongator and coarse-level candidates are

$$T_0 = \begin{bmatrix} 0.5774 & -0.7071 & 0.0000 & 0.0000 \\ 0.5774 & 0.0000 & 0.0000 & 0.0000 \\ 0.5774 & 0.7071 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.5774 & -0.7071 \\ 0.0000 & 0.0000 & 0.5774 & 0.0000 \\ 0.0000 & 0.0000 & 0.5774 & 0.7071 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1.7321 & 1.7321 \\ 0.0000 & 1.4142 \\ 1.7321 & 6.9282 \\ 0.0000 & 1.4142 \end{bmatrix}. \tag{2.3}$$

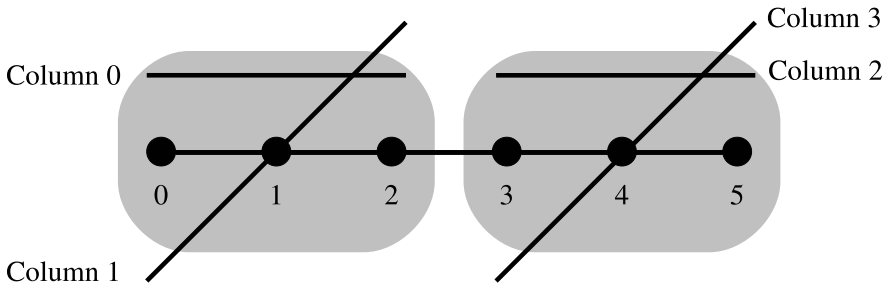

Figure 2.3: Unsmoothed coarse basis functions of the tentative prolongator T_0.

The four coarse-level basis functions of T_0 are illustrated in Figure 2.3.

At this point, one option is to take $P_n = T_n$ and construct an unsmoothed aggregation multigrid hierarchy. For instance, the two level method $A_0 = A, A_1 = T_0^T A T_0$ has the property that error components which lie in the span of the candidate vectors over each aggregate are completely annihilated by a multigrid cycle. Unfortunately, this property alone does not provide optimal multigrid convergence and therefore a *prolongation smoother* is applied to the tentative prolongator before forming the next coarser level.

Prolongation smoothing, represented by `SmoothProlongator` in Algorithm 2.2, improves the approximation of algebraically smooth errors by the final prolongator P_n. Smoothing is accomplished by computing the sparse matrix-matrix product $(I - S_n A_n)T_n$, where S_n determines the kind of smoothing. Possible implementations for S_n include Richardson $S_n = \omega I$, Jacobi $S_n = \omega \operatorname{diag}(A)^{-1}$, and polynomial smoothers $S_n = p(A)$ [3]. Prolongation smoothing on the column $t_j = T(:, j)$ of the tentative prolongator (i.e. a coarse basis function), which results in the smoothed column $(I - S_n A_n)t_j$, is equivalent to applying one iteration of the stationary relaxation method $t_j \Leftarrow t_j + S_n(b - A_n t_j)$ on the linear system $At_j = 0$. In other words, the coarse basis functions of the smoothed prolongator P_n are simply relaxed versions of the tentative basis functions.

As a general principle, smoother coarse basis functions better approximate algebraically smooth error and therefore provide superior multigrid convergence. However, prolongator smoothing broadens the support of each coarse basis function, which in turn contributes additional nonzeros (or fill) to the smoothed prolongator P_n and the coarse level system $P_n^T A_n P_n$. Therefore, the total *work per digit* (WPD) of accuracy of the resultant multigrid cycle will not necessarily improve with more aggressive prolongator smoothing.

In SA, the procedures `Restrict` and `Interpolate` of the generic multigrid cycle in Algorithm 2.1 are implemented by P_k^T and P_k respectively. Algorithm 2.3 demonstrates this specialized multigrid cycle. Finally, the hierarchy construction and solve phases are combined in Algorithm 2.4. Provided with a matrix of near-nullspace candidates $B = [b_0, b_1, \ldots, b_{K-1}]$, procedure `sa_solver`

Algorithm 2.3: `amg_cycle`$(A_0, \ldots, A_N, P_0, \ldots, P_{N-1}, x_l, b_l)$

```
1  if l = N
2      return Solve(A_l, b_l)
3  else
4      x_l ⇐ Presmooth(A_l, x_l, b_l)
5      r_l ⇐ b_l − A_l x_l
6      b_{l+1} ⇐ P_k^T r_l
7      x_{l+1} ⇐ 0
8      x_{l+1} ⇐ amg_cycle(A_0, ..., A_N, P_0, ..., P_{N−1}, x_{l+1}, b_{l+1})
9      x_l ⇐ x_l + P_k x_{l+1}
10     x_l ⇐ Postsmooth(A_l, x_l, b_l)
11     return x_l
12 end
```

Algorithm 2.4: `sa_solver`$(A, B, x, b, tolerance)$

```
1  A_0, ..., A_N, P_0, ..., P_{N−1} ⇐ sa_hierarchy(A, B)
2  while ||b − Ax|| > tolerance
3      x ⇐ amg_cycle(A_0, ..., A_N, P_0, ..., P_{N−1}, x, b)
4  end
5  return x
```

solves the linear system $Ax = b$ to a given residual tolerance.

2.1.1 Adaptive Smoothed Aggregation

The construction of a standard smoothed aggregation hierarchy, embodied by Algorithm 2.2, requires a matrix of near-nullspaces candidates. When this information is unavailable, the adaptive smoothed aggregation (αSA) method [14] computes a set of candidates automatically.

The αSA algorithm develops a set of candidates by applying several iterations of the method to the system $Ax = \mathbf{0}$ with a random initial vector. The result $\tilde{x}$, if nonzero, is a vector that is not effectively reduced by the existing multigrid cycle. Introducing $\tilde{x}$ into the range of interpolation – i.e. augmenting the columns of $B = [B, \tilde{x}]$ – ensures that the new cycle annihilates $\tilde{x}$. The new cycle is then applied to the system $Ax = 0$ to discover additional candidate vectors until a limit on the number of candidates is reached or the method is found to effectively reduce all error modes.

Since the αSA algorithm adds considerable cost to the hierarchy construction phase, it is only used when a set of suitable near-nullspace candidates is not available. Furthermore, the number of candidate vectors required for effective multigrid cycling performance generally exceeds the number of vectors in the standard method. For instance, whereas SA requires three candidates for 2D linear elasticity problems, αSA requires as many as 5 to recover the same rate

of convergence [14]. Despite the additional computational burden, the adaptive smoothed aggregation methodology is useful and constitutes a true black box method.

2.2 Terminology

When comparing the performance of algebraic multigrid methods there are several metrics to consider. The *operator complexity* (OC) is a measure of the size of the complete hierarchy relative to the size of the initial matrix. Given the sequence of matrices $A_0, A_1, \ldots, A_N$ in the multigrid hierarchy, the operator complexity is computed as $\sum_{i=0}^{N} \text{nnz}(A_i)/\text{nnz}(A_0)$, where nnz is the *number of nonzeros* in the matrix. Operator complexity is an approximate measure of the memory and setup cost of a hierarchy. Since operator complexity varies by problem, discretization, and dimension it is difficult to define an ideal value. Generally speaking, operator complexities less than 2.0 are acceptable, while considerably larger values are less desireable and potentially problematic.

Related to operator complexity is the *work per digit of accuracy* (WPD) of the solver. As the name suggests, WPD measures the number of floating point operations required to reduce the error by a factor of 10. Since the true error is not typically available, we measure the reduction in the residual norm $||b - Ax||$. The WPD is calculated as

$$\frac{\log(0.1)}{\log(C)} \sum_{i=0}^{N} R_i/R \tag{2.4}$$

where R_i is the total number of floating point operations used during relaxation on level i, R is the cost of a single relaxation pass on the finest level, and C is the average convergence ratio[3] of the method. For instance, if a multigrid V-cycle with a single iteration of Jacobi relaxation during pre- and post-smoothing is applied to a hierarchy with an operator complexity of 1.5 to produce a convergence ratio of 0.1, then the WPD is 3.0. If, instead, two iterations of Jacobi relaxation during pre- and post-smoothing on the same hierarchy produced a convergence ratio of 0.1, then the corresponding WPD is 6.0.

[3]Geometric mean of the per-iteration convergence ratios.

Chapter 3

Discrete Differential Forms

Discrete differential forms arise in scientific disciplines ranging from computational electromagnetics to computer graphics. Examples include stable discretizations of the eddy-current problem [46, 12, 4], topological methods for sensor network coverage [16], visualization of complex flows [36, 41], surface parameterization [25], and the design of vector fields on meshes [21]. This Chapter provides an overview of discrete forms and their properties through concrete examples. We refer the interested reader to [28, 10, 17] for additional background information and motivation.

3.1 Example

Consider the three element simplicial mesh depicted in Figure 3.1 with vertices, edges, and triangles enumerated as shown. Note that the edges and triangles are *oriented*. In general, an oriented p-simplex (or *face*) is represented by an ordered tuple of indices $(s_0, s_1, \ldots, s_p)$. In this example, edge 4 has the unique representation $(2, 3)$ while triangle 1 is equivalently represented by either $(1, 2, 3)$ or $(3, 1, 2)$ or $(2, 3, 1)$. The representation of an oriented simplex is unique up to an even permutation of its indices.

In the smooth theory the *exterior derivative* $\boldsymbol{d}$ generalizes the vector calculus operations $\boldsymbol{\nabla}\cdot$, $\boldsymbol{\nabla}$, and $\boldsymbol{\nabla}\times$ to manifolds in non-Euclidean spaces. Given a mesh with oriented faces, the discrete k-form derivative, denoted $\mathbb{D}_k$, is constructed to implement a discrete analog of the exterior derivative. For this example, the

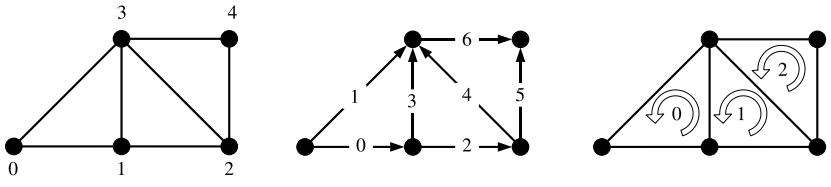

Figure 3.1: Enumeration of nodes (left), oriented edges (center), and oriented triangles(right) for a simple triangle mesh.

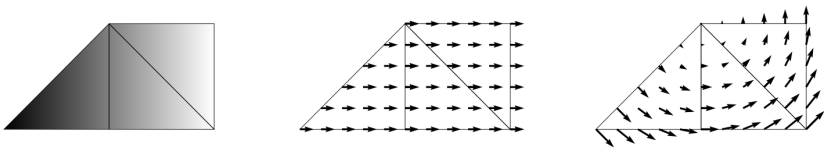

Figure 3.2: Forms $\mathcal{I}_0\alpha^0$, $\mathcal{I}_1\mathbb{D}_0\alpha^0$, and $\mathcal{I}_1\beta^1$ where $\mathcal{I}$ denotes Whitney interpolation. The left and center figures illustrate property 3.4. Whether the derivative is applied before or after interpolation, the result is the same.

sparse matrices

$$\mathbb{D}_{-1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbb{D}_0 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, \tag{3.1}$$

$$\mathbb{D}_1 = \begin{bmatrix} 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 \end{bmatrix}, \qquad \mathbb{D}_2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \tag{3.2}$$

implement the discrete k-form derivative operators. Each row of the k-th derivative operator records the relative orientation between the boundary of a $k+1$-dimensional face and its k-dimensional subfaces. In the boundary of triangle 0, edges 0 and 3 occur with positive orientation while edge 1 appears with negative orientation. Hence it follows that $\mathbb{D}(0,0) = \mathbb{D}(0,3) = 1$ and $\mathbb{D}(0,1) = -1$. In fact, the discrete derivative is defined to be the adjoint of the corresponding boundary operator[28], i.e. $< \mathbb{D}_u, v > = < u, \partial_{k+1} v >$. By convention $\mathbb{D}_{-1}$ and $\mathbb{D}_2$ are included to complete the sequence.

Discrete derivatives act on *discrete k-forms* which are represented as column vectors with real values for each of the k-simplices in the mesh. The entries of a discrete k-form, denoted α^k, represent *integrated* quantities. To discretize a continuous scalar field, or 0-form, point samples at each of the vertices are computed to determine the entries of the discrete 0-form. Likewise, the values of a discrete 1-form are computed by integrating a continuous 1-form along the edges of the mesh. This method of discretizing smooth k-forms is called the *de Rham* map [45, 10, 28] and is denoted $\mathcal{R}$.

The reverse process, interpolating discrete k-forms over a mesh, is denoted

$\mathcal{I}$. Interpolated fields corresponding to k-forms $\alpha^0 = [0, 1, 2, 1, 2]^T$, the gradient $\mathbb{D}_0\alpha^0 = [1, 1, 1, 0, -1, 0, 1]$, and another 1-form $\beta^1 = [1, 0, 1, 0, 0, 1, 0]$ are shown in Figure 3.2. In this example, Whitney forms [45, 13] have been used to define $\mathcal{I}$. Similar interpolation methods exist for hexahedral [11] and polyhedral elements [24].

3.2 Properties

Discretizations that mimic the structure of the exterior calculus, so-called *mimetic methods*, retain properties of the smooth theory [28, 10]. In particular, the *de Rham complex* formed by the discrete derivative operators,

$$0 \xrightarrow{\mathbb{D}_{-1}} \Omega_d^0 \xrightarrow{\mathbb{D}_0} \Omega_d^1 \xrightarrow{\mathbb{D}_1} \dots \quad \Omega_d^N \xrightarrow{\mathbb{D}_N} 0, \tag{3.3}$$

is a chain complex, i.e. $\text{img}(\mathbb{D}_k) \subset \ker(\mathbb{D}_{k+1})$ or equivalently $\mathbb{D}_{k+1}\mathbb{D}_k = 0$. The corresponding result in the smooth theory, i.e. $\boldsymbol{d}_{k+1}\boldsymbol{d}_k = \mathbf{0}$, is responsible for the vector calculus identities $\boldsymbol{\nabla} \cdot \boldsymbol{\nabla} \times = \mathbf{0}$ and $\boldsymbol{\nabla} \times \boldsymbol{\nabla} = \mathbf{0}$ in three-dimensional Euclidean space. Mimicking this property of the smooth theory is essential when considering discretizations of the eddy-current problem [4].

The interpolated forms $\mathcal{I}_0\alpha^0$ and $\mathcal{I}_1\mathbb{D}_0\alpha^0$ depicted in Figure 3.2 illustrate commutativity of the interpolation and derivative operations. This relationship is depicted as

$$\begin{array}{ccc} \Omega^k & \xrightarrow{\boldsymbol{d}_k} & \Omega^{k+1} \\ \uparrow{\scriptstyle \mathcal{I}_k} & & \uparrow{\scriptstyle \mathcal{I}_{k+1}} \\ \Omega_d^k & \xrightarrow{\mathbb{D}_k} & \Omega_d^{k+1} \end{array} \tag{3.4}$$

where Ω^k and Ω_d^k denote the spaces of smooth differential k-forms and discrete k-forms respectively. When computing the bilinear form

$$< \boldsymbol{d}_k\mathcal{I}_k u^k, \boldsymbol{d}_k\mathcal{I}_k v^k >, \tag{3.5}$$

we exploit the relationship in Diagram 3.4 to substitute the simpler expression

$$< \mathcal{I}_{k+1}\mathbb{D}_k u^k, \mathcal{I}_{k+1}\mathbb{D}_k v^k > = \mathbb{D}_k^T \mathbb{M}_{k+1}\mathbb{D}_k, \tag{3.6}$$

using the mass matrix $\mathbb{M}_{k+1} = < \mathcal{I}_{k+1}, \mathcal{I}_{k+1} >$.

The de Rham map $\mathcal{R}_k$, which discretizes smooth k-forms, also commutes

with the exterior derivative:

$$\begin{array}{ccc} \Omega^k & \xrightarrow{\boldsymbol{d}_k} & \Omega^{k+1} \\ \downarrow{\scriptstyle \mathcal{R}_k} & & \downarrow{\scriptstyle \mathcal{R}_{k+1}} \\ \Omega_d^k & \xrightarrow{\mathbb{D}_k} & \Omega_d^{k+1} \end{array} \tag{3.7}$$

One practical benefit of Diagram 3.7 is the ability to discretize forms using the gradient of a lower-dimensional form. For instance, the discrete 1-form corresponding to $\mathbf{1}dx$ may be computed by sampling the x-coordinate at each of the vertices of a mesh (a discrete 0-form) and then applying $\mathbb{D}_0$ to the result.

3.3 Focus and Applications

In this thesis we consider we consider solving discretizations of problems of the form,

$$\boldsymbol{\delta d}\alpha^k = \beta^k, \tag{3.8}$$

where $\boldsymbol{d}$ denotes the exterior derivative and $\boldsymbol{\delta}$ the codifferential relating smooth k-forms α and β. For $k = 0, 1, 2$, $\boldsymbol{\delta d}$ is also expressed as $\boldsymbol{\nabla}\cdot\boldsymbol{\nabla}$, $\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times$, and $\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot$ respectively. We refer to operator $\boldsymbol{\delta d}$ generically as a Laplacian, although it does not correspond to the Laplace-de Rham operator $\boldsymbol{\Delta} = \boldsymbol{d\delta} + \boldsymbol{\delta d}$ except for the case $k = 0$. We assume that (3.8) is discretized with mimetic first-order elements such as Whitney forms [45, 13] on simplicial meshes or the analog on hexahedral [11] or polyhedral elements [24].

In general we use $\mathcal{I}_k$ to denote the map from discrete k-forms (cochains) to their respective finite elements. Such discretizations give rise to a discrete exterior k-form derivative $\mathbb{D}_k$ and discrete k-form innerproduct

$$\mathbb{M}_k(i,j) = <\mathcal{I}_k e_i, \mathcal{I}_k e_j>, \tag{3.9}$$

which allows implementation of (3.8) in weak-form as

$$\mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k x = b, \tag{3.10}$$

under the additional assumption that $\boldsymbol{d}$ and $\mathcal{I}$ satisfy Diagram 3.4. We consider solving (3.10) on structured or unstructured meshes of arbitrary dimension and element type, provided that the discretization satisfies the aforementioned properties.

In Chapter 7 we discuss computing Hodge decompositions of discrete k-forms with the proposed methods. The Hodge decomposition is a fundamental tool in both pure and applied mathematics that exposes topological information through differential forms. For example, the two harmonic 1-forms shown in

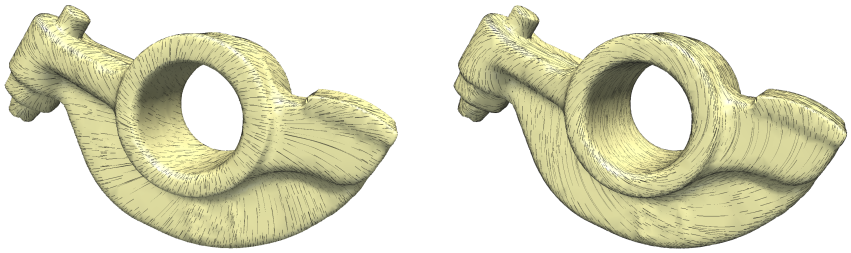

Figure 3.3: The two harmonic 1-forms of a rocker arm surface mesh.

Figure 3.3 exist because the manifold has genus 1. When computing discrete Hodge decompositions, we encounter linear systems with matrices of the form

$$\mathbb{D}_k^T \mathbb{D}_k, \qquad \mathbb{D}_k \mathbb{D}_k^T, \qquad \mathbb{D}_k \mathbb{M}_{k+1}^{-1} \mathbb{D}_k^T, \tag{3.11}$$

in addition to $\mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k$ of Equation 3.10.

The efficient solution of discrete k-form Laplacians has substantial utility in computational topology, where, for instance, sufficient conditions on the coverage of sensor networks reduce to the discovery of harmonic forms on the simplicial Rips complex [16]. In this application, we encounter the combinatorial Laplacian

$$\Delta_k = \mathbb{D}_{k-1} \mathbb{D}_{k-1}^T + \mathbb{D}_k^T \mathbb{D}_k, \tag{3.12}$$

and determine whether it has a non-trivial nullspace. We explore efficient solvers for this problem in Chapter 8.

Chapter 4

Chain Complex Method

This chapter develops the *chain complex method* (cSA) for solving k-form problems. The cSA method creates a hierarchy of progressively coarser levels that preserve the structure of the de Rham complex (cf. Figure 3.3). Specifically, the discrete derivative operators within each level of the hierarchy form a *chain complex* (Section 3.2). Furthermore, interpolation operators that act between levels of the hierarchy and commute with the coarse- and fine-level discrete derivatives are constructed. The interpolation operators serve as the tentative prolongators within the smoothed aggregation methodology. Finally, we demonstrate how cSA is used to efficiently solve linear systems with the discrete k-form Laplacian operators

$$\mathbb{D}_k^T \mathbb{D}_k, \qquad \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k, \qquad \mathbb{D}_k \mathbb{D}_k^T, \tag{4.1}$$

that appear in the discrete Hodge decompositions discussed in Chapter 7.

4.1 Background

There is significant interest in efficient solution methods for Maxwell's eddy-current problem

$$\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \vec{E} + \sigma \vec{E} = \vec{f}. \tag{4.2}$$

In particular, recent approaches focus on multilevel methods for both structured and unstructured meshes [27, 5, 37, 29]. Scalar multigrid performs poorly on edge element discretizations of (4.2) since error modes that lie in the kernel of $\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times$ are not effectively damped by standard relaxation methods. Fortunately, the problematic modes are easily identified by the range of the discrete gradient operator $\mathbb{D}_0$, and an appropriate hybrid smoother [27, 5] can be constructed. An important property of these multigrid methods is commutativity

between coarse and fine finite element spaces. The relationship is described as

$$\begin{array}{ccc} \Omega_d^0 & \xrightarrow{\mathbb{D}_0} & \Omega_d^1 \\ \uparrow{\scriptstyle P_0} & & \uparrow{\scriptstyle P_1} \\ \widehat{\Omega}_d^0 & \xrightarrow{\widehat{\mathbb{D}}_0} & \widehat{\Omega}_d^1 \end{array} \tag{4.3}$$

where $\widehat{\Omega}_d^k$ is the space of coarse discrete k-forms, $\widehat{\mathbb{D}}_0$ the coarse gradient operator, and P_0 and P_1 are the nodal and edge prolongation operators respectively. Combining (4.3) with (3.4) yields the same result for the corresponding fine and coarse finite element spaces.

Reitzinger and Schöberl [37] describe an algebraic multigrid method for solving (4.2) on unstructured meshes. In their method, property (4.3) is maintained by choosing nodal aggregates and using these aggregates to obtain compatible edge aggregates. The nodal and edge aggregates give rise to piecewise-constant prolongators P_0 and P_1, which can be smoothed to achieve better multigrid convergence rates [29] while retaining property (4.3).

The method we present can be viewed as a natural extension of Reitzinger and Schöberl's work from 1-forms to general k-forms. Commutativity of the coarse and fine de Rham complexes is maintained for all k-forms, and their associated finite element spaces $\mathcal{I}_k\Omega_d^k \subset \Omega^k$. The relationship is described by

$$\begin{array}{ccccccccc} \Omega_d^0 & \xrightarrow{\mathbb{D}_0} & \Omega_d^1 & \xrightarrow{\mathbb{D}_1} & \Omega_d^2 & \cdots & \Omega_d^k & \xrightarrow{\mathbb{D}_k} & \Omega_d^{k+1} \\ \uparrow{\scriptstyle \mathcal{P}_0} & & \uparrow{\scriptstyle \mathcal{P}_1} & & \uparrow{\scriptstyle \mathcal{P}_2} & & \uparrow{\scriptstyle \mathcal{P}_k} & & \uparrow{\scriptstyle \mathcal{P}_{k+1}} \\ \widehat{\Omega}_d^0 & \xrightarrow{\widehat{\mathbb{D}}_0} & \widehat{\Omega}_d^1 & \xrightarrow{\widehat{\mathbb{D}}_1} & \widehat{\Omega}_d^2 & \cdots & \widehat{\Omega}_d^k & \xrightarrow{\widehat{\mathbb{D}}_k} & \widehat{\Omega}_d^{k+1} \end{array} \tag{4.4}$$

where $\mathcal{P}_k$ denotes either the tentative prolongator P_k or smoothed prolongator $\mathcal{S}_k P_k$.

4.2 Complex Coarsening

In this section we describe the construction of tentative prolongators P_k and coarse operators $\widehat{\mathbb{D}}_k$ which satisfy (4.4). In practice, the two-level commutativity depicted in (4.4) is extended recursively for use in a multilevel method. Also, it is important to note that when solving (3.10) for a specific k, it is not necessary to coarsen the entire complex.

As in [37], we presume the existence of a nodal aggregation algorithm which produces a piecewise-constant tentative prolongator P_0. This procedure, called `aggregate_nodes` in Algorithm 4.1, is fulfilled by either Smoothed Aggregation [44] or a graph partitioner on matrices $\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$ or $\mathbb{D}_0^T\mathbb{D}_0$. Ideally, the

Algorithm 4.1: `coarsen_complex`$(\mathbb{D}_{-1}, \mathbb{D}_0, \ldots, \mathbb{D}_N)$

```
P_0 ⇐ aggregate_nodes(D_0, ...)
for k = 0 to N − 1
    P_{k+1} ⇐ induced_aggregates(P_k, D_k, D_{k+1})
    D̂_k ⇐ (P_{k+1}^T P_{k+1})^{-1} P_{k+1}^T D_k P_k
end
D̂_{-1} ⇐ P_0^T D_{-1}
D̂_N ⇐ D_N P_N
return P_0, P_1, ..., P_N and D̂_{-1}, D̂_0, ..., D̂_N
```

nodal aggregates are contiguous and have a small number of interfaces with other aggregates.

4.3 Induced Aggregates

The key concept in [37], which we apply and extend here, is that nodal aggregates *induce* edge aggregates; we denote P_1 as the resulting edge aggregation operator. As depicted in Figure 4.1, a coarse edge exists between two coarse nodal aggregates when any fine edge joins them. Multiple fine edges between the same two coarse nodal aggregates interpolate from a common coarse edge with weight 1 or -1 depending on their orientation relative to the coarse edge. The coarse nodes and coarse edges define a coarse derivative operator $\widehat{\mathbb{D}}_0$ which satisfies diagram (4.3).

We now restate the previous process in an algebraic manner that generalizes to arbitrary k-forms. Given P_0 as before, form the product $\overline{\mathbb{D}} = \mathbb{D}_0 P_0$ which relates coarse nodes to fine edges. Observe that each row of $\overline{\mathbb{D}}$ corresponds to a fine edge and each column to a coarse node. Notice that the i-th row of $\overline{\mathbb{D}}$ is zero when the endpoints of fine edge i lie within the same nodal aggregate. Conversely, the i-th row of $\overline{\mathbb{D}}$ is nonzero when the endpoints of fine edge i lie in different nodal aggregates. Furthermore, when two nonzero rows are equal up to sign (i.e. linearly dependent), they interpolate from a common coarse edge.

Therefore, the procedure of aggregating edges reduces to computing sets of linearly dependent rows in $\overline{\mathbb{D}}$. Each set of dependent rows yields a coarse edge and thus a column of P_1. In the general case, sets of dependent rows in $\overline{\mathbb{D}} = \mathbb{D}_k P_k$ are identified and used to produce P_{k+1}. The process can be repeated to coarsen the entire de Rham complex. Alternatively, the coarsening can be stopped at a specific $k < N$. In Section 4.6 we discuss the coarse derivative operator $\widehat{\mathbb{D}}_k \Leftarrow (P_{k+1}^T P_{k+1})^{-1} P_{k+1}^T \mathbb{D}_k P_k$ and show that it satisfies diagram (4.4).

Intuitively, linear dependence between rows in $\overline{\mathbb{D}} = \mathbb{D}_k P_k$ indicates redundancy created by operator P_k. Aggregating dependent rows together removes redundancy from the output of $\overline{\mathbb{D}}$ and compresses the remaining degrees of freedom into a smaller set of variables. By construction, the tentative prolongators

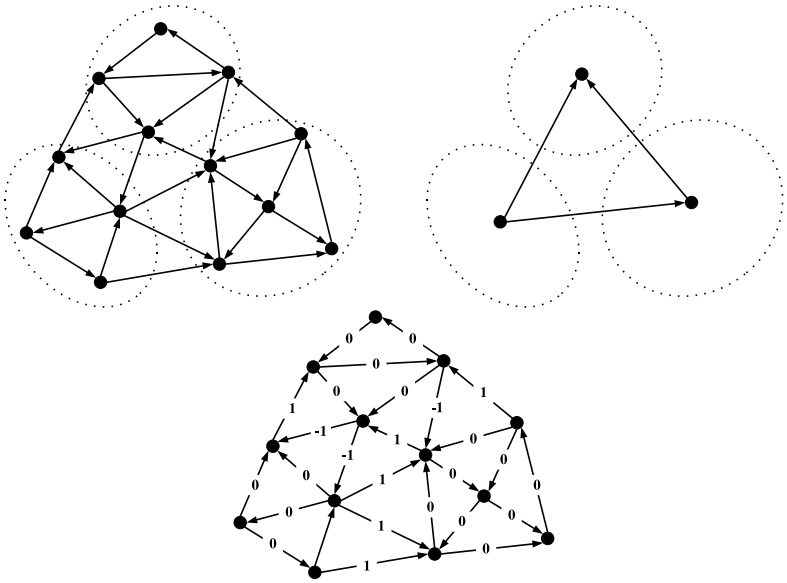

Figure 4.1: Nodal aggregates (upper left) determine coarse edges (upper right) through the algorithm `induced_aggregates`. Fine edges crossing between node aggregates interpolate from the corresponding coarse edge with weight 1 or -1 depending on their relative orientation. Edges contained within an aggregate do not correspond to any coarse edge and receive weight 0. These weights are determined by lines 10-13 of `induced_aggregates`.

Algorithm 4.2: `induced_aggregates`$(P_k, \mathbb{D}_k, \mathbb{D}_{k+1})$

```
1  D̄ ⇐ D_k P_k
2  G ⇐ D_{k+1}^T D_{k+1}
3  V ⇐ {}
4  n ⇐ 0
5
6  for i in rows(D̄) such that D̄(i,:) ≠ 0
7     if i ∉ V
8        A_n ⇐ dependent_rows(G, D̄, i)
9        for j ∈ A_n
10          if D̄(i,:) = D̄(j,:)
11             P_{k+1}(j,n) ⇐ 1
12          else
13             P_{k+1}(j,n) ⇐ −1
14          end
15       end
16       n ⇐ n + 1
17       V ⇐ V ∪ A_n
18    end
19 end
20 return P_{k+1}
```

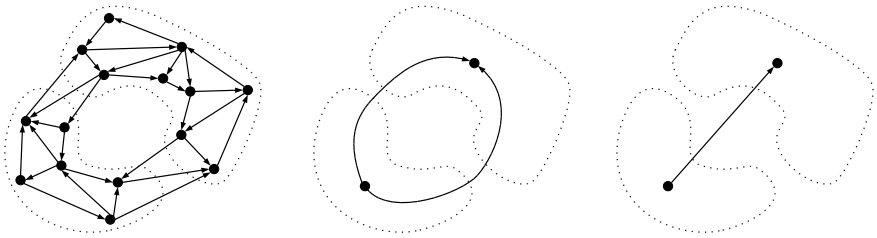

Figure 4.2: Example where contiguous (center) and non-contiguous (right) aggregation differs. Contiguous aggregates are reflected through our choice of $\mathbb{G}$ defined in `induced_aggregates` and later used in `dependent_rows`.

have full column rank and satisfy

$$\mathcal{R}(\mathbb{D}_k P_k) \subset \mathcal{R}(P_{k+1}) \tag{4.5}$$

where $\mathcal{R}(A)$ denotes the range of matrix A. Note that property (4.5) is clearly necessary to satisfy diagram (4.4).

Using disjoint sets of dependent rows A_0,A_1,..., the function `induced_aggregates` constructs the aggregation operator P_{k+1} described above. Non-zero entry $P_{k+1}(i,j)$ indicates membership of the i-th row of $\overline{\mathbb{D}}$—i.e. the i-th $k+1$-dimensional element—to the j-th aggregate A_j.

4.4 Computing Aggregates

For a given row index i, the function `dependent_rows` constructs a set of rows that are linearly dependent to $\overline{\mathbb{D}}(i,:)$. In the matrix graph of $\mathbb{G}$, a nonzero entry $\mathbb{G}(i,j)$ indicates that the $k+1$-dimensional elements with indices i and j are *upper-adjacent* [33]. In other words, i and j are both faces of some $k+2$ dimensional element. For example, two edges in a simplicial mesh are upper-adjacent if they belong to the same triangle. All linearly dependent rows that are adjacent in the matrix graph of $\mathbb{G}$ are aggregated together. This construction ensures that the aggregates produced by `dependent_rows` are contiguous. As shown in Figure 4.2, such aggregates are more natural than those that result from aggregating all dependent rows together (i.e. using $\mathbb{G} = \overline{\mathbb{D}}\ \overline{\mathbb{D}}^T$).

4.5 Example

In this section we describe the steps of our algorithm applied to the three element simplicial mesh depicted in Figure 3.1. Matrices $\mathbb{D}_{-1}, \mathbb{D}_0, \mathbb{D}_1$, and $\mathbb{D}_2$, shown in Section 3.1, are first computed and then passed to `coarsen_complex`. The externally-defined procedure `aggregate_nodes` is then called to produce the

Algorithm 4.3: `dependent_rows`$(\mathbb{G}, \overline{\mathbb{D}}, i)$

```
Q ⇐ {i}
A ⇐ {i}
while Q ≠ {}
   j ⇐ pop(Q)
   Q ⇐ Q \ {j}
   for k such that 𝔾(j,k) ≠ 0
      if k ∉ A and 𝔻̄(i,:) = ±𝔻̄(k,:)
         A ⇐ A ∪ {k}
         Q ⇐ Q ∪ {k}
      end
   end
end
return A
```

piecewise-constant nodal aggregation operator

$$P_0 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

whose corresponding aggregates are shown in Figure 4.3. At this stage of the procedure, a more general nodal problem $\mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0$ may be utilized in determining the coarse aggregates. Next, `induced_aggregates` is invoked with arguments $P_0, \mathbb{D}_0, \mathbb{D}_1$, and the sparse matrix

$$\overline{\mathbb{D}} = \mathbb{D}_0 P_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \tag{4.7}$$

is constructed. Recall from Section 4.3 that the rows of $\overline{\mathbb{D}}$ are used to determine the induced edge aggregates. The zero rows of $\overline{\mathbb{D}}$, namely rows 0, 1, and 3, correspond to interior edges, which is confirmed by Figure 4.3. Linear dependence between rows 2 and 4 indicates that edges 2 and 4 have common coarse endpoints, with the difference in sign indicating opposite orientations.

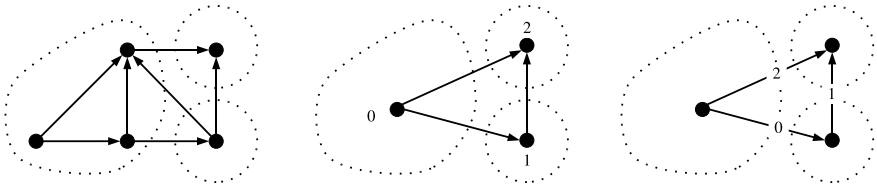

Figure 4.3: Original mesh with nodal aggregates (left), coarse nodes (center), and coarse edges (right).

For each non-zero and un-aggregated row of $\overline{\mathbb{D}}$, `dependent_rows` traverses

$$\mathbb{G} = \mathbb{D}_1^T \mathbb{D}_1 = \begin{bmatrix} 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 2 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix}, \tag{4.8}$$

to find dependent rows among upper-adjacent edges. In this case, edges 3 and 4 are upper-adjacent to 2, however only row 4 in $\overline{\mathbb{D}}$ is linearly dependent to row 2 in $\overline{\mathbb{D}}$. Rows 5 and 6 of $\overline{\mathbb{D}}$ are not linearly dependent to any other rows, thus forming single aggregates for edges 5 and 6. The resulting aggregation operator

$$P_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.9}$$

is then used to produce the coarse discrete derivative operator

$$\widehat{\mathbb{D}}_0 = (P_1^T P_1)^{-1} P_1^T \mathbb{D}_0 P_0 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \tag{4.10}$$

for the mesh in Figure 4.3. Subsequent iterations of the algorithm produce

operators

$$P_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \widehat{\mathbb{D}}_1 = (P_2^T P_2)^{-1} P_2^T \mathbb{D}_1 P_1 = \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}, \quad \widehat{\mathbb{D}}_2 = \mathbb{D}_2 P_2 = \begin{bmatrix} 0 \end{bmatrix}, \tag{4.11}$$

which complete the coarse de Rham complex.

4.6 Commutativity

We now prove tentative prolongators $P_0, P_1, \ldots, P_K$ and coarse derivative operators $\widehat{\mathbb{D}}_0, \widehat{\mathbb{D}}_1, \ldots, \widehat{\mathbb{D}}_K$ produced by Algorithm 4.1 satisfy commutative diagram (4.4). The result is summarized by the following theorem.

Theorem 1. *Let $P_k : \widehat{\Omega}_d^k \to \Omega_d^k$ denote the discrete k-form prolongation operators with the following properties*

$$P_{k+1} \quad \text{has full column rank} \tag{4.12i}$$

$$\mathcal{R}(\mathbb{D}_k P_k) \subset \mathcal{R}(P_{k+1}) \tag{4.12ii}$$

$$\widehat{\mathbb{D}}_k \Leftarrow (P_{k+1}^T P_{k+1})^{-1} P_{k+1}^T \mathbb{D}_k P_k \tag{4.12iii}$$

Then, diagram (4.4) holds. That is,

$$\mathbb{D}_k P_k = P_{k+1} \widehat{\mathbb{D}}_k \tag{4.13}$$

Proof. Since P_{k+1} has full column rank, the pseudoinverse is given by

$$P_{k+1}^+ = (P_{k+1}^T P_{k+1})^{-1} P_{k+1}^T. \tag{4.14}$$

Recall that for an arbitrary matrix A the pseudoinverse satisfies $AA^+A = A$. Furthermore, $\mathcal{R}(\mathbb{D}_k P_k) \subset \mathcal{R}(P_{k+1})$ implies that $\mathbb{D}_k P_k = P_{k+1}\mathbb{X}$ for some matrix $\mathbb{X}$. Combining these observations,

$$\begin{aligned} P_{k+1}\widehat{\mathbb{D}}_k &= P_{k+1} P_{k+1}^+ \mathbb{D}_k P_k, \\ &= P_{k+1} P_{k+1}^+ P_{k+1} \mathbb{X}, \\ &= P_{k+1} \mathbb{X}, \\ &= \mathbb{D}_k P_k \end{aligned}$$

□

Since Algorithm 4.1 meets assumptions (4.12i),(4.12ii), and (4.12iii) it follows that diagram (4.4) is satisfied. Also, assuming disjoint aggregates, the matrix $(P_{k+1}^T P_{k+1})$ appearing in (4.14) is a diagonal matrix, so its inverse is easily computed.

4.7 Chain Complex

The de Rham complex formed by the fine-level discrete derivative operators,

$$0 \xrightarrow{\mathbb{D}_{-1}} \Omega_d^0 \xrightarrow{\mathbb{D}_0} \Omega_d^1 \xrightarrow{\mathbb{D}_1} \dots \quad \Omega_d^N \xrightarrow{\mathbb{D}_N} 0 \quad (4.15)$$

is a chain complex, i.e. $\text{img}(\mathbb{D}_k) \subset \ker(\mathbb{D}_{k+1})$ or equivalently $\mathbb{D}_{k+1}\mathbb{D}_k = 0$. A natural question to ask is whether the coarse complex retains this property. As argued in Section 4.6, $\mathbb{D}_k P_k = P_{k+1}\mathbb{X}$ for some matrix $\mathbb{X}$, therefore it follows

$$\begin{aligned}
\widehat{\mathbb{D}}_{k+1}\widehat{\mathbb{D}}_k &= P_{k+2}^+ \mathbb{D}_{k+1} P_{k+1} P_{k+1}^+ \mathbb{D}_k P_k, \\
&= P_{k+2}^+ \mathbb{D}_{k+1} P_{k+1} P_{k+1}^+ P_{k+1} \mathbb{X}, \\
&= P_{k+2}^+ \mathbb{D}_{k+1} P_{k+1} \mathbb{X}, \\
&= P_{k+2}^+ \mathbb{D}_{k+1} \mathbb{D}_k P_k, \\
&= 0,
\end{aligned}$$

since $\mathbb{D}_{k+1}\mathbb{D}_k = 0$ by assumption. From diagram (3.4) we infer the same result for the associated finite element spaces.

4.8 Smoothed Prolongators

While the tentative prolongators $P_0, P_1, \dots$ produced by `coarsen_complex` commute with $\mathbb{D}_k$ and give rise to a coarse chain complex, their piecewise-constant nature leads to suboptimal multigrid scaling [37, 29]. In Smoothed Aggregation [44], the tentative prolongator P is smoothed to produce another prolongator $\mathcal{P} = \mathcal{S}P$ with superior approximation characteristics. We consider prolongation smoothers of the form $\mathcal{S} = (I - \mathbb{S}A)$. Possible implementations include Richardson $\mathbb{S} = \omega I$, Jacobi $\mathbb{S} = \omega \,\text{diag}(A)^{-1}$, and polynomial $\mathbb{S} = p(A)$ [3].

Smoothed prolongation operators are desirable, but straightforward application of smoothers to each of $P_0, P_1, \dots$ violates commutativity. The solution proposed in [29] smooths P_0 and P_1 with *compatible* smoothers $\mathcal{S}_0, \mathcal{S}_1$ such that commutativity of the smoothed prolongators $\mathcal{P}_0, \mathcal{P}_1$ is maintained, i.e. $\mathbb{D}_0\mathcal{P}_0 = \mathcal{P}_1\widehat{\mathbb{D}}_0$. In the following theorem we generalize this result to arbitrary k.

Theorem 2. *Given discrete k-form prolongation operators P_k satisfying (4.12i), (4.12ii), and (4.12iii), let $\mathcal{P}_k : \widehat{\Omega}_d^k \to \Omega_d^k$ denote the smoothed discrete k-form prolongation operators with the following properties*

$$\mathcal{P}_k = \mathcal{S}_k P_k \quad (4.16\text{i})$$

$$\mathcal{S}_0 = (I - \mathbb{S}_0 \mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0) \quad (4.16\text{ii})$$

$$\mathcal{S}_k = (I - \mathbb{S}_k \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k - \mathbb{D}_{k-1}\mathbb{S}_{k-1}\mathbb{D}_{k-1}^T \mathbb{M}_k) \quad \text{for k} > 0 \quad (4.16\text{iii})$$

where $\mathbb{S}_k$ defines the type of prolongation smoother. Then, diagram (4.4) holds.

That is,

$$\mathbb{D}_k \mathcal{P}_k = \mathcal{P}_{k+1} \widehat{\mathbb{D}}_k \tag{4.17}$$

Proof. First, if

$$\mathbb{D}_k \mathcal{S}_k = \mathcal{S}_{k+1} \mathbb{D}_k \tag{4.18}$$

then

$$\begin{aligned}
\mathcal{P}_{k+1}\widehat{\mathbb{D}}_k &= \mathcal{S}_{k+1} P_{k+1} \widehat{\mathbb{D}}_k \\
&= \mathcal{S}_{k+1} P_{k+1} (P_{k+1}^T P_{k+1})^{-1} P_{k+1}^T \mathbb{D}_k P_k \\
&= \mathcal{S}_{k+1} \mathbb{D}_k P_k \\
&= \mathbb{D}_k \mathcal{S}_k P_k \\
&= \mathbb{D}_k \mathcal{P}_k
\end{aligned}$$

Therefore, it suffices to show that (4.18) holds for all k. For $k = 0$ we have

$$\begin{aligned}
\mathcal{S}_1 \mathbb{D}_0 &= (I - \mathbb{S}_1 \mathbb{D}_1^T \mathbb{M}_2 \mathbb{D}_1 - \mathbb{D}_0 \mathbb{S}_0 \mathbb{D}_0^T \mathbb{M}_1) \mathbb{D}_0 \\
&= (\mathbb{D}_0 - \mathbb{S}_1 \mathbb{D}_1^T \mathbb{M}_2 \mathbb{D}_1 \mathbb{D}_0 - \mathbb{D}_0 \mathbb{S}_0 \mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0) \\
&= (\mathbb{D}_0 - \mathbb{D}_0 \mathbb{S}_0 \mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0) \\
&= \mathbb{D}_0 (I - \mathbb{S}_0 \mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0) \\
&= \mathbb{D}_0 \mathcal{S}_0
\end{aligned}$$

and for all $k > 1$ we have

$$\begin{aligned}
\mathcal{S}_{k+1} \mathbb{D}_k &= (I - \mathbb{S}_{k+1} \mathbb{D}_{k+1}^T \mathbb{M}_{k+2} \mathbb{D}_{k+1} - \mathbb{D}_k \mathbb{S}_k \mathbb{D}_k^T \mathbb{M}_{k+1}) \mathbb{D}_k \\
&= (\mathbb{D}_k - \mathbb{S}_{k+1} \mathbb{D}_{k+1}^T \mathbb{M}_{k+2} \mathbb{D}_{k+1} \mathbb{D}_k - \mathbb{D}_k \mathbb{S}_k \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k) \\
&= (\mathbb{D}_k - \mathbb{D}_k \mathbb{S}_k \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k) \\
&= (\mathbb{D}_k - \mathbb{D}_k \mathbb{S}_k \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k - \mathbb{D}_k \mathbb{D}_{k-1} \mathbb{S}_{k-1} \mathbb{D}_{k-1}^T \mathbb{M}_k) \\
&= \mathbb{D}_k (I - \mathbb{S}_k \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k - \mathbb{D}_{k-1} \mathbb{S}_{k-1} \mathbb{D}_{k-1}^T \mathbb{M}_k) \\
&= \mathbb{D}_k \mathcal{S}_k
\end{aligned}$$

which completes the proof of (4.17). □

On subsequent levels, the coarse innerproducts $\widehat{\mathbb{M}}_k = \mathcal{P}_k^T \mathbb{M}_k \mathcal{P}_k$ and derivatives $\widehat{\mathbb{D}}_k$ replace $\mathbb{M}_k$ and $\mathbb{D}_k$ in the definition of $\mathcal{S}_k$. As shown below, the Galerkin product $\widehat{A}_k = \mathcal{P}_k^T A_k \mathcal{P}_k$ can also be written in terms of the coarse operators.

$$\begin{aligned}
\widehat{A}_k &= \mathcal{P}_k^T A_k \mathcal{P}_k \\
&= \mathcal{P}_k^T \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k \mathcal{P}_k \\
&= \widehat{\mathbb{D}}_k^T \mathcal{P}_{k+1}^T \mathbb{M}_{k+1} \mathcal{P}_{k+1} \widehat{\mathbb{D}}_k \\
&= \widehat{\mathbb{D}}_k^T \widehat{\mathbb{M}}_{k+1} \widehat{\mathbb{D}}_k
\end{aligned}$$

4.9 Extensions

Note that condition (4.5) permits some freedom in our choice of aggregates. For instance, in restricting ourselves to contiguous aggregates we have slightly enriched the range of P_{k+1} beyond what is necessary. Provided that P_{k+1} already satisfies (4.5), additional coarse basis functions can be introduced to better approximate low-energy modes. As in Smoothed Aggregation, these additional columns of P_{k+1} can be chosen to exactly interpolate given near-nullspace vectors [44].

So far we have only discussed coarsening the cochain complex (4.4). It is worth noting that `coarsen_complex` works equally well on the chain complex formed by the mesh boundary operators $\partial_k = \mathbb{D}_{k-1}^T$,

$$0 \xleftarrow{\mathbb{D}_{-1}^T} \Omega_d^0 \quad \ldots \xleftarrow{\mathbb{D}_{N-2}^T} \Omega_d^{N-1} \xleftarrow{\mathbb{D}_{N-1}^T} \Omega_d^N \xleftarrow{\mathbb{D}_N^T} 0, \quad (4.19)$$

by simply reversing the order of the complex,

$$\mathbb{D}_{-1}, \mathbb{D}_{,0}, \ldots, \mathbb{D}_N \Rightarrow \mathbb{D}_N^T, \mathbb{D}_{N-1}^T, \ldots, \mathbb{D}_{-1}. \quad (4.20)$$

In this case `aggregate_nodes` will aggregate top-level elements, such as the triangles in Figure 3.1. Intuitively, ∂_k acts like a derivative operator that maps k-cochains to $(k+1)$-cochains, however one typically refers to these as k-chains rather than cochains [28].

4.10 Numerical Results

We have applied the proposed method to a number of structured and unstructured problems. In all cases, a multigrid V(1,1)-cycle is used as a preconditioner to conjugate gradient iteration. Unless stated otherwise, a symmetric Gauss-Seidel sweep is used during pre- and post-smoothing stages. Iteration on the positive-semidefinite systems,

$$\mathbb{D}_k^T \mathbb{D}_k, \qquad \mathbb{D}_k \mathbb{D}_k^T, \qquad \mathbb{D}_k^T \mathbb{M}_{k+1} \mathbb{D}_k, \quad (4.21)$$

proceeds until the relative residual is reduced by 10^{-10}. The matrix $\mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0$ corresponds to a Poisson problem with pure-Neumann boundary conditions. Similarly, $\mathbb{D}_1^T \mathbb{M}_2 \mathbb{D}_1$ is an eddy-current problem (4.2) with $\sigma = 0$. As explained in Chapter 7, matrices (4.21) arise in discrete Hodge decompositions.

The multigrid hierarchy extends until the number of unknowns falls below 500, at which point a pseudoinverse is used to perform the coarse level solve.

The tentative prolongators are smoothed *twice* with a Jacobi smoother

$$\mathcal{S} = I - \frac{4}{3\lambda_{max}} \operatorname{diag}(A)^{-1} A \tag{4.22}$$

$$\mathcal{P} = \mathcal{S}\mathcal{S}P \tag{4.23}$$

where λ_{max} is an upper bound on the spectral radius of $\operatorname{diag}(A)^{-1}A$. When zero or near zero values appear on the diagonal of the Galerkin product $\mathcal{P}^T A \mathcal{P}$, the corresponding rows and columns are zeroed and ignored during smoothing. We discuss this choice of prolongation smoother in Section 4.11.

Tables 4.1 and 4.2 show the result of applying the proposed method to regular quadrilateral and hexahedral meshes of increasing size. In both cases, the finite element spaces described in [11] are used to produce the innerproducts $\mathbb{M}_k$. The systems are solved with a random initial value for x. Since the matrices are singular, the solution x is an arbitrary null vector. Column labels are explained as follows:

- 'Grid' - dimensions of the quadrilateral/hexahedral grid
- 'Convergence'- average convergence factor
- 'WPD' - work per digit of accuracy [1]
- 'OC' - operator complexity
- 'Levels' - number of levels in the multigrid hierarchy

For each k, the algorithm exhibits competitive convergence factors while maintaining low operator complexity. Together, the work per digit-of-accuracy remains bounded as the problem size increases.

In Table 4.3, numerical results are presented for the unstructured tetrahedral mesh depicted in Figure 4.4. As with classical algebraic multigrid methods, performance degrades in moving from a structured to an unstructured tessellation. However the decrease in performance for the scalar problems $\mathbb{D}_0^T \mathbb{D}_0$ and $\mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0$ is less significant than that of the other problems.

4.11 Prolongation Smoother Comparison

On the nonscalar problems considered, we found second degree prolongation smoothers noticeably more efficient than first degree prolongation smoothers. While additional smoothing operations generally improve the convergence rate of Smoothed Aggregation methods, this improvement is typically offset by an increase in operator complexity and therefore the resultant work per digit of accuracy is not improved. However, there is an important difference between the tentative prolongators in the scalar and nonscalar problems. In the scalar case,

[1] Excluding the cost of conjugate gradient iteration.

System	Grid	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_0^T\mathbb{D}_0$	250^2	63,001	0.075	5.817	1.636	4
	500^2	251,001	0.100	6.644	1.661	4
	1000^2	1002,001	0.063	5.617	1.686	5
$\mathbb{D}_1^T\mathbb{D}_1$	250^2	125,500	0.096	5.919	1.506	4
	500^2	501,000	0.103	6.187	1.527	5
	1000^2	2,002,000	0.085	5.773	1.545	5
$\mathbb{D}_0\mathbb{D}_0^T$	250^2	125,500	0.124	6.751	1.530	4
	500^2	501,000	0.133	7.040	1.542	5
	1000^2	2,002,000	0.094	6.049	1.553	5
$\mathbb{D}_1\mathbb{D}_1^T$	250^2	62,500	0.063	5.467	1.641	4
	500^2	250,000	0.063	5.477	1.664	4
	1000^2	1,000,000	0.063	5.620	1.687	5
$\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$	250^2	63,001	0.043	4.142	1.415	4
	500^2	251,001	0.055	4.547	1.432	4
	1000^2	1,002,001	0.041	4.175	1.448	5
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	250^2	125,500	0.095	5.893	1.506	4
	500^2	501,000	0.103	6.187	1.527	5
	1000^2	2,002,000	0.085	5.773	1.545	5

Table 4.1: Two-dimensional scaling results.

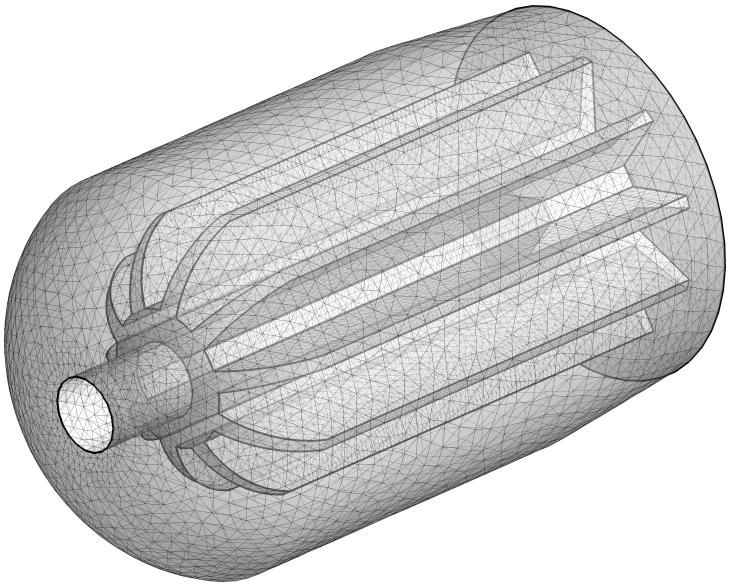

Figure 4.4: Titan IV rocket mesh.

System	Grid	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_0^T\mathbb{D}_0$	25^3	17,576	0.120	5.508	1.268	3
	50^3	132,651	0.151	6.333	1.300	3
	100^3	1,030,301	0.105	5.550	1.358	4
$\mathbb{D}_1^T\mathbb{D}_1$	25^3	50,700	0.192	7.233	1.296	3
	50^3	390,150	0.216	8.066	1.342	4
	100^3	3,060,300	0.208	8.300	1.415	4
$\mathbb{D}_2^T\mathbb{D}_2$	25^3	48,750	0.188	6.371	1.156	3
	50^3	382,500	0.218	7.135	1.180	3
	100^3	3,030,000	0.267	8.488	1.217	4
$\mathbb{D}_0\mathbb{D}_0^T$	25^3	50,700	0.287	9.194	1.246	3
	50^3	390,150	0.391	12.113	1.235	4
	100^3	3,060,300	0.323	10.204	1.252	4
$\mathbb{D}_1\mathbb{D}_1^T$	25^3	48,750	0.187	7.630	1.389	3
	50^3	382,500	0.264	9.703	1.403	4
	100^3	3,030,000	0.194	8.172	1.455	4
$\mathbb{D}_2\mathbb{D}_2^T$	25^3	15,625	0.089	4.957	1.302	3
	50^3	125,000	0.102	5.318	1.318	3
	100^3	1,000,000	0.103	5.543	1.368	4
$\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$	25^3	17,576	0.037	3.291	1.178	3
	50^3	132,651	0.053	3.763	1.200	3
	100^3	1,030,301	0.038	3.495	1.241	4
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	25^3	50,700	0.097	4.674	1.184	3
	50^3	390,150	0.113	5.128	1.214	4
	100^3	3,060,300	0.088	4.790	1.264	4
$\mathbb{D}_2^T\mathbb{M}_3\mathbb{D}_2$	25^3	48,750	0.188	6.371	1.156	3
	50^3	382,500	0.223	7.243	1.180	3
	100^3	3,030,000	0.265	8.440	1.217	4

Table 4.2: Three-dimensional scaling results.

System	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_0^T\mathbb{D}_0$	84,280	0.073	4.588	1.304	3
$\mathbb{D}_1^T\mathbb{D}_1$	554,213	0.378	13.197	1.391	4
$\mathbb{D}_2^T\mathbb{D}_2$	920,168	0.366	10.868	1.186	4
$\mathbb{D}_0\mathbb{D}_0^T$	554,213	0.236	14.601	2.289	4
$\mathbb{D}_1\mathbb{D}_1^T$	920,168	0.390	11.708	1.197	4
$\mathbb{D}_2\mathbb{D}_2^T$	450,235	0.370	9.662	1.043	3
$\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$	84,280	0.144	6.197	1.304	3
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	554,213	0.518	20.765	1.483	4
$\mathbb{D}_2^T\mathbb{M}_3\mathbb{D}_2$	920,168	0.348	10.357	1.187	4

Table 4.3: Performance of cSA on the tetrahedral rocket mesh.

System	Grid	Degree	Percent Zero	Convergence	WPD	OC
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	250^2	0	66.8	0.697	42.255	1.123
		1	66.8	0.357	14.774	1.123
		2	22.9	0.096	8.379	1.506
		3	0.4	0.063	9.515	2.084
		4	0.0	0.063	10.188	2.250
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	50^3	0	67.6	0.567	25.043	1.034
		1	66.5	0.290	11.497	1.035
		2	8.8	0.096	7.460	1.214
		3	0.3	0.063	9.011	1.577
		4	0.0	0.063	9.074	1.632
$\mathbb{D}_2^T\mathbb{M}_3\mathbb{D}_2$	50^3	0	89.63	0.549	23.670	1.034
		1	89.63	0.382	14.753	1.034
		2	63.93	0.214	10.304	1.180
		3	23.77	0.122	9.203	1.481
		4	6.48	0.098	8.348	1.487
		5	2.07	0.089	10.267	1.953

Table 4.4: Comparison of prolongation smoothers.

all degrees of freedom are associated with a coarse aggregate and therefore the tentative prolongator has no zero rows. As described in Section 4.5, the tentative prolongator for nonscalar problems has zero rows for elements contained in the interior of a nodal aggregate. In the nonscalar case, additional smoothing operations incorporate a greater proportion of these degrees of freedom into the range of the final prolongator.

The influence of higher degree prolongation smoothers on solver performance is reported in Table 4.4. Column 'Degree' records the degree d of the prolongation smoother $\mathcal{P} = \mathcal{S}^d P$ while 'Percent Zero' reflects the percentage of zero rows in the first level prolongator. As expected the operator complexity increases with smoother degree. However, up to a point, this increase is less significant than the corresponding reduction in solver convergence. Second degree smoothers exhibit the best efficiency in both instances of the problem $\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$ and remain competitive with higher degree smoothers in the last test. Since work per digit figures exclude the cost of constructing multigrid transfer operators, these higher degree smoothers may be less efficient in practice.

Chapter 5

k-Form Basis Method

The chain complex method (cSA) described in Chapter 4 constructs a multigrid hierarchy by aggregating nodes of a scalar problem and then computing the *induced* higher-dimensional aggregates. Since the cSA coarsening procedure operates directly on discrete derivative operators, problems discretized on both structured and unstructured meshes, and their dual meshes, are handled uniformly. Furthermore, coarser levels form chain complexes and, through interpolation, commute with the finer levels.

The results of the numerical study in Section 4.10 indicate that cSA is more effective for problems discretized on structured meshes than those on unstructured meshes. In particular, when a non-trivial innerproduct[1] is introduced to an unstructured problem, solver convergence degrades. This example highlights the primary deficiency of the chain complex approach: the aggregation process does not consider the spatial embedding of the elements. For instance, in the 1-form problem $\mathbb{D}_1^T \mathbb{M}_2 \mathbb{D}_1$ any two edge elements that belong to the same face are considered for membership in the same aggregate. Due to the locality of aggregates, the members of an aggregate typically represent similar fields. In the case of discrete 1-forms or edge elements, the constituent edges ought to "point" in the same direction. The structured mesh depicted in Figure 5.1 illustrates an ideal instance for the cSA method. Indeed, with the nodal aggregates shown, the coarse complex produced by cSA is natural geometric coarsening of the fine-level mesh.

While the cSA procedure does consider the orientation of faces, it is only in a topological sense. For example, perpendicular edges are allowed to aggregate together. On the other hand, when cSA is applied to structured meshes, the shape of the initial nodal aggregates tends to favor the creation of meaningful higher-dimensional coarse-level aggregates. Unfortunately, this property of cSA on structured meshes does not carry over to the unstructured case.

In this Chapter we introduce an algorithm to specifically address the aforementioned limitations of cSA. The proposed method, which we call the *k-form basis Method* (kSA), applies the smoothed aggregation methodology to discrete k-form problems. The primary distinction between the cSA and kSA methods is that the former maintains the differential structure throughout the multigrid hi-

[1] e.g. $\mathbb{M}_k$ determined by Whitney interpolation

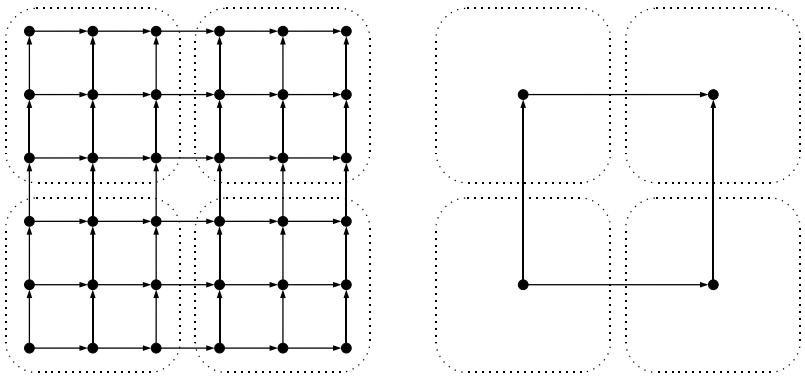

Figure 5.1: With appropriate nodal aggregates, such as those at left, cSA coarsening produces similarly structured coarse complexes. In this example, three fine edges with the same orientation extend from each coarse edge (right).

erarchy while the latter ensures that certain k-forms are accurately represented at all levels.

5.1 Discrete k-Form Bases

Recall that construction of the Smoothed Aggregation (SA) hierarchy (cf. Section 2.1) requires a problem-dependent set of near-nullspace candidate vectors. Given an aggregation of the fine-level degrees of freedom, the candidates are used to ensure that near-nullspace modes are well-approximated by the range of the tentative prolongators.

The cSA method deviates from the "standard" smoothed aggregation approach in that the tentative prolongator is computed directly, effectively combining `Aggregate` and `FitCandidates` steps of Algorithm 2.2 into a single procedure. Subsequents steps of the cSA setup algorithm, namely prolongator smoothing and formation of the Galerkin operator $P^T AP$, follow the standard SA methodology. While coarser grids of the cSA hierarchy maintain various properties, they are not constructed to capture any particular set of low-energy modes.

As an alternative, we choose a specific set of near nullspace candidates to be reproduced exactly. Inspired by the work of [8], the kSA method applies this approach to general k-form problems. By suppling SA with a *k-form basis*, we ensure their representation on coarser levels. These basis vectors, like the rigid-body modes in linearized elasticity, are computed directly from the mesh representation.

5.1.1 Discrete 0-form Bases

When solving the Poisson problem with smoothed aggregation, the constant vector $b_0 = [1, 1, \ldots, 1]^T$ suffices to capture the near-kernel. Since the matrix

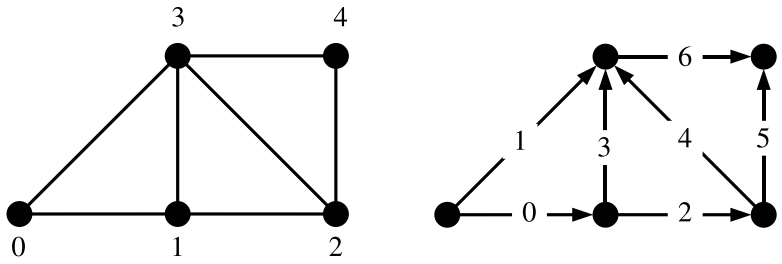

Figure 5.2: Enumeration of vertices (left) and oriented edges (right) for a simple triangle mesh.

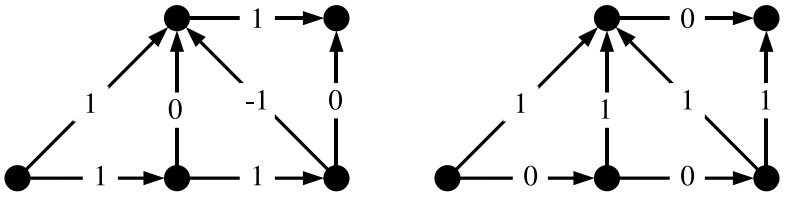

Figure 5.3: Applying the de Rham map $\mathcal{R}$ to dx, i.e. integrating the continuous 1-form dx along the edges of the mesh, produces the coefficients in the left figure. The right figure displays the same procedure applied to dy.

$\mathbb{D}_0^T \mathbb{M}_1 \mathbb{D}_0$ of the discrete 0-form problem is simply a scalar Laplacian operator, the constant vector again suffices. In the context of k-form bases, the constant function $\mathbf{1}$ is a basis for 0-forms on a manifold: i.e. any 0-form can be described as the product of a scalar function $f(x)$ and $\mathbf{1}$.

5.1.2 Discrete 1-form Bases

In two dimensional Euclidean space, a 1-form basis is spanned by dx and dy, which are identified with vector fields in the directions of the x and y axes respectively. Likewise, dx, dy, and dz constitute a 1-form basis in three dimensional Euclidean space. In general, a manifold parameterized with D coordinates has D distinct 1-form basis elements. Recall from Section 3.1 that discrete 1-forms (1-cochains) are represented by a vector of scalar values, with one value for each mesh edge. Therefore, a discrete 1-form basis for a manifold with D coordinates is represented by a matrix B with dimension $N_E \times D$, where N_E is the number of mesh edges.

Consider the problem of computing a 1-form basis for the two-dimensional mesh shown in Figure 5.2. The discrete 1-form corresponding to the continuous field dx is obtained by computing the line integral of dx along each edge of the mesh and storing the results in a column vector. The discrete 1-form for dy is computed in a similar fashion. In Euclidean space, this is equivalent to computing the line integral of the corresponding coordinate vector fields along

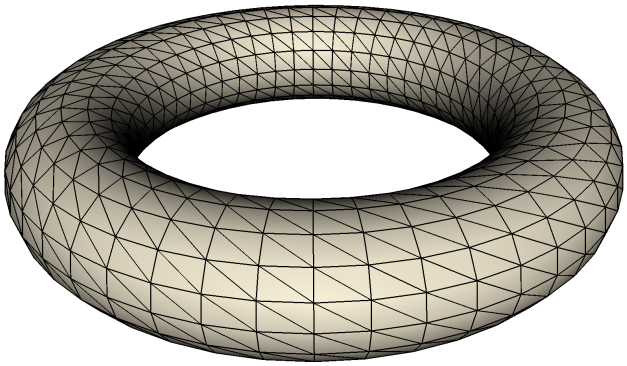

Figure 5.4: Simplicial discretization of the torus.

each mesh edge. Figure 5.3 illustrates the coefficients associated with each of the mesh. We use the enumeration of edges shown in Figure 5.2 to arrange each set of coefficients into a column vector. Column vectors are then assembled into a matrix

$$B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ -1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{5.1}$$

which constitutes a complete 1-form basis for this mesh. In kSA, this matrix provides the near-nullspace candidates for the hierarchy construction process.

The previous process is a specific application of the de Rham map $\mathcal{R}$, introduced in Section 3.1, to discretized continuous 1-forms. In the general case, we apply the de Rham map $\mathcal{R}$ to each of the smooth 1-form basis functions to obtain a discrete 1-form basis. Although we are primarily concerned with discrete manifolds embedded in two and three-dimensional Euclidean space, the method generalizes further. For instance, the surface of a torus is (locally) parameterized by coordinates u and v. Discretizing du and dv on the torus mesh (Figure 5.4) provides a valid 1-form basis for the kSA method. Since the torus mesh is also embedded in three-dimensional Euclidean space, the 1-form basis consisting of dx, dy, and dz provides another alternative. We explore this example in Section 5.5.

While one can apply the previously mentioned approach to compute discrete 1-form bases, a computationally simpler method exists for meshes embedded in Euclidean space. Consider again the mesh in Figure 5.2 whose vertex coordi-

nates are specified by the matrix

$$V = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 0 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}. \tag{5.2}$$

Note that the columns of V can also be viewed as discrete versions of the scalar fields x and y. Specifically, the first and second columns of V are $\mathcal{R}x$ and $\mathcal{R}y$. Exploiting the commutative relationship depicted in Diagram 3.7 allows one to compute $\mathcal{R}dx$ as $\mathbb{D}_0\mathcal{R}x$, or the complete 1-form basis $B = \mathbb{D}_0 V$.

5.1.3 Discrete 2-form Bases

The 2-forms $dx \wedge dy$, $dx \wedge dz$, and $dy \wedge dz$ span a basis in three-dimensional Euclidean space. Here, $\wedge$ denotes the *wedge product* of exterior calculus [1, 22]. The definition $\wedge : \Omega^{k_1} \times \Omega^{k_2} \rightarrow \Omega^{k_1+k_2}$ implies that wedge product is used to construct higher-dimensional forms from two lower-dimensional forms. Since k-forms are meant to be integrated over k-dimensional spaces, the wedge product provides a way to integrate area (2-form) and volume (3-form) beginning with the notion of length (1-form). For instance, integrating the 2-form $dx \wedge dy$ over a subset of the Euclidean plane yields the region's area.

In Euclidean 3-space, a 2-form measures flux density. For example, integrating $dx \wedge dy$ over a two-dimensional surface yields the net flux through the surface of the unit vector field in the z-direction. Similarly, $dx \wedge dz$, and $dy \wedge dz$ measure flux density in the $-y$ and x directions respectively. Intuitively, $dx \wedge dy$ represents an infinitesimal square region swept out by 1-forms dx and dy.

As an alternative to computing the directly 2-form basis directly, one may apply the approach advocated for 1-forms, i.e. discretizing a lower-dimensional form whose derivative is the desired result. For example, the discrete 1-form $\mathcal{R}xdy$ is a potential field for the 2-form $\mathcal{R}dx \wedge dy$ which is computed $\mathbb{D}_1\mathcal{R}xdy$.

5.1.4 General k-form Bases

In the general case, a k-form basis on a D-dimensional manifold has $\binom{D}{k}$ elements. The two methods we have discussed, direct application of the de Rham map $\mathcal{R}$ and use of a discrete potential $\mathbb{D}_{k-1}\mathcal{R}$, also apply to the general case. Alternatively, if a discrete wedge product [28] is available, a k-form basis can be constructed recursively from a 1-form basis. Specifically, once the D distinct discrete 1-form basis elements are computed, each of the $\binom{D}{k}$ k-form basis elements are computed by wedging the k constituent 1-forms together.

5.1.5 Dual Meshes

Sections 5.1.1-5.1.4 outlined the construction of bases for k-form problems of the form $\mathbb{D}_k^T\mathbb{D}_k$ and $\mathbb{D}_k^T\mathbb{M}_{k+1}\mathbb{D}_k$. The unknowns of these problems are associated with the nodes, edges, facets, and volumes of the *primal mesh*. In contrast, inputs to operators of the form $\mathbb{D}_k\mathbb{D}_k^T$ are identified with elements of the *dual mesh*.

We construct k-form bases for dual problems by applying $\mathcal{R}$ to the *barycentric dual* of the mesh. For instance, the barycentric dual of a tetrahedron is a vertex at the barycenter of the tetrahedron. Likewise, the dual of a triangular face in a tetrahedral mesh is an edge that joins the barycenters of the tetrahedra on either side of the face. Structured quadrilateral and hexahedral meshes admit a similar geometric dual mesh. Boundary elements, such as the surface facets of a 3D mesh, do not have proper dual mesh counterparts and are assigned the value 0 in the discrete basis.

Our use of the dual mesh is motivated by the geometric duals that appear in the development of *covolume methods* [28, 34]. Covolume methods use the elements in the primal and *circumcentric dual* meshes to construct a discrete Hodge star operator which is then used to transfer quantities between the primal and dual meshes and to define the innerproduct $\mathbb{M}_k$. For our domain of interest, the barycentric dual is more appropriate than the circumcentric dual since the latter is applicable only to *well-centered* meshes, which are uncommon, while the former exists for all meshes. Indeed, the creation of well-centered meshes remains an open problem [43].

5.2 Coarse Basis Functions

In this section we compare and contrast the coarse basis functions produced by the cSA and kSA methods. In cSA, the aggregation of higher-dimensional k-forms is induced by nodal aggregates on the 0-form problem. Nodal aggregates are computed with the standard aggregation algorithm [44] using the scalar Laplacian matrix $\mathbb{D}_0^T\mathbb{D}$. Figure 5.5 illustrates example nodal aggregates and induced coarse edges produced by cSA coarsening in the context of a 1-form problem. For this example, the coarse-level problem has three degrees of freedom (DoFs), one for each coarse edge. As described in Section 4.3, the relative orientation between coarse and fine-level edges determines the coefficients of each coarse basis function, which form the columns of the tentative interpolation operator. Figure 5.6 depicts these tentative coarse basis functions.

Since the cSA hierarchy has a one-to-one relationship between coarse aggregates and coarse-level degrees of freedom, the coarser grids resemble coarser discretizations of the same domain. While intuitive, this property needlessly restricts the choice of coarse basis functions. Specifically, cSA does not utilize the support for multiple DoFs per aggregate in smoothed aggregation. As discussed

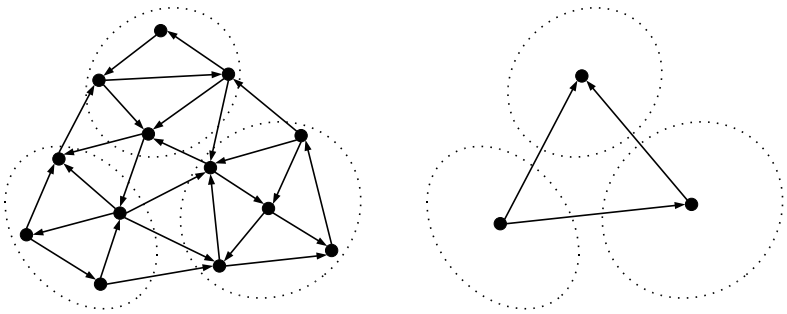

Figure 5.5: Examples of node and edge aggregates produced by the cSA method. In this case, three coarse edges are induced by the nodal aggregation.

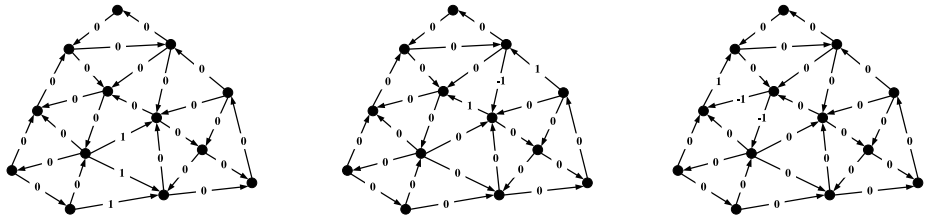

Figure 5.6: The tentative prolongator of the cSA method has three basis functions, corresponding to each of the coarse edges. Fine-level edges that cross between nodal aggregates interpolate from a coarse edge with a value determined by their relative orientations.

in Section 2.1, the use of several near-nullspace candidates is instrumental in solving vector-valued problems such as linearized elasticity with smoothed aggregation. In such applications, the relationship between coarse and fine-level unknowns is less direct than that of cSA.

A potential improvement to cSA is to by develop more than one coarse hierarchy. Since different nodal aggregations give rise to different higher-dimensional aggregates, using several cSA coarsenings at each level of the hierarchy tends to improve interpolation. Unfortunately this approach greatly increase the cost of the cSA method without ensuring that smooth modes are well-approximated.

In contrast to cSA, the kSA method aggregates the degrees of freedom of k-form problems directly and does not maintain the chain complex property on coarser levels. Although we postpone a full discussion of the aggregation algorithm itself until Chapter 6, we remark here that the sparsity structure of k-form problems is poorly suited to the standard SA coarsening. For now we presume that the unknowns are partitioned into disjoint aggregates similar to the partition of edge elements in Figure 5.8.

Despite having similar spatial partitions, the kSA basis functions shown in Figure 5.9 are qualitatively different from the cSA basis functions illustrated in Figure 5.6. Indeed, the cSA basis functions have limited support which leaves many fine-level edges outside the range of the tentative prolongators. In

contrast, the kSA tentative prolongator extends to all fine edges. While prolongator smoothing has the effect of broadening the support region of coarse basis functions, it is an imprecise and potentially expensive means of extending interpolation to all fine-level degrees of freedom. As demonstated in Section 4.11, many fine degrees of freedom only appears in the tentative prolongator after several prolongation smoothing steps. However, since prolongation smoothing is a global process, the resulting multigrid hierarchy has considerably higher complexity.

Another qualitative difference between the cSA and kSA methods is that the former retains a similar structure on coarser levels while the latter does not. Specifically, the second level of a cSA hierarchy resembles a coarser mimetic discretization of the same domain while the second level of a kSA hierarchy resembles a coarser discretization using nodal finite elements. As illustrated by Figure 5.7, subsequent levels of the kSA hierarchy have the same structure as other vector-valued problems (e.g. linearized elasticity). In light of this quality, the kSA hierarchy can be viewed as a one-level transformation to a nodal problem followed by a traditional smoothed aggregation hierarchy.

In this regard, the kSA method is similar to the work of Bochev et al. [9]. Like the cSA method, their approach uses nodal aggregation to determine coarse-level basis functions. However, unlike cSA the coarse basis functions are chosen to fit a set of candidate vectors. Rather than forming disjoint edge aggregates, the support of the basis functions extends to all edges with at least one endpoint in each nodal aggregate. Representative aggregates and coarse basis functions are shown in Figures 5.10 and 5.11 respectively. Here, the weighting of edges that join separate nodal aggregates can be viewed as a simplified form of prolongator smoothing which ensures that the 1-form basis still lies within the range of interpolation.

5.3 Proposed Method

Recall the SA hierarchy construction procedure detailed in Algorithm 2.2. With the k-form operator A and the discrete k-form basis discussed in Section 5.1 providing B, the matrix of near-nullspace candidates, the requirements of the construction algorithm are satisfied. However, for reasons explained in Chapter 6, we replace the standard aggregation algorithm, denoted `Aggregate` in Algorithm 2.2, with Lloyd aggregation on the *first level* of coarsening.

Other approaches, such as cSA and the method of Bochev et al. [9], use nodal aggregation methods to aggregate higher-dimensional discrete forms. In contrast, kSA aggregates discrete k-forms directly with Lloyd aggregation. Compared to cSA, which must also coarsen all discrete operators of dimension less than k, this approach avoids a considerable amount of work.

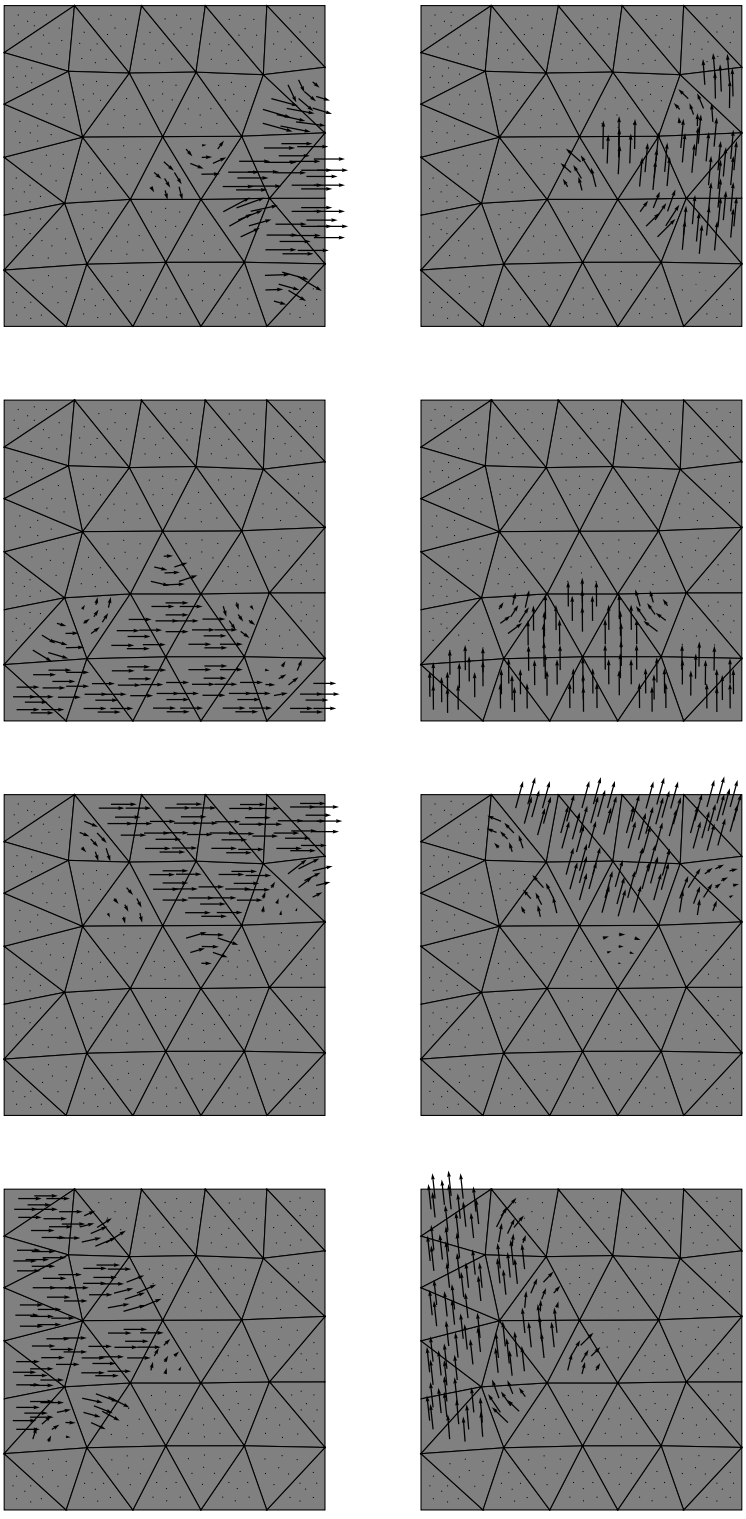

Figure 5.7: Tentative coarse basis functions produced by the kSA method.

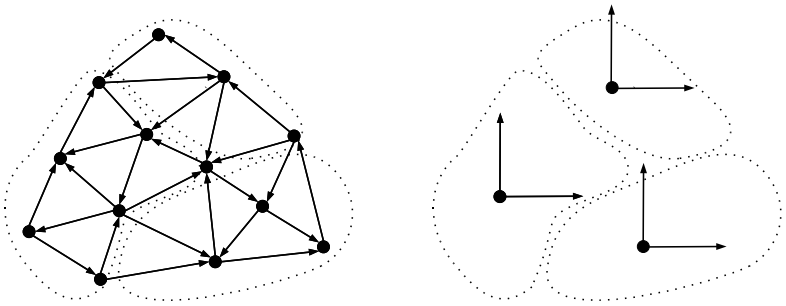

Figure 5.8: Example edge aggregates (left) used by the kSA method. Each aggregate supports two basis functions corresponding to the 1-forms dx and dy for a total of six coarse-level degrees of freedom.

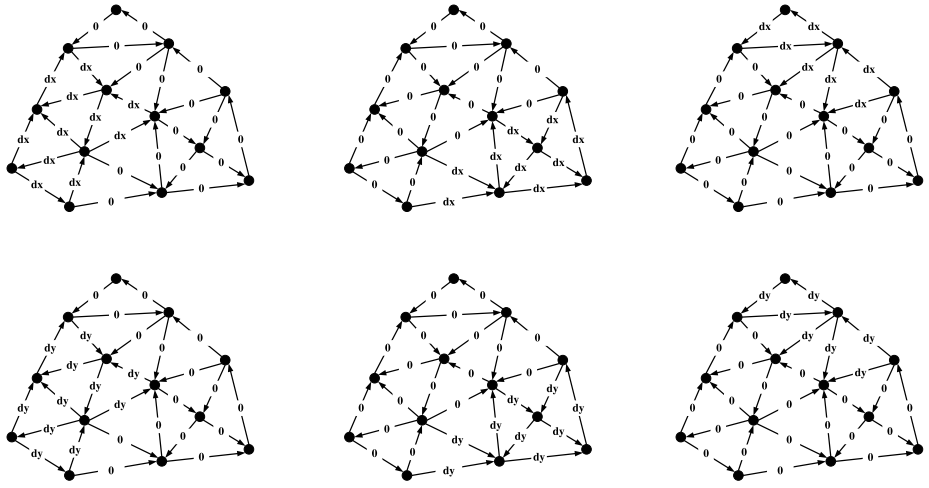

Figure 5.9: Discrete 1-form basis functions dx and dy are restricted to each edge aggregate to determine the coarse basis functions. The value of dx or dy on an edge may be found by applying the de Rham map to the corresponding continuous 1-form. Differencing the x and y coordinates at the endpoints of each edge yields an equivalent result.

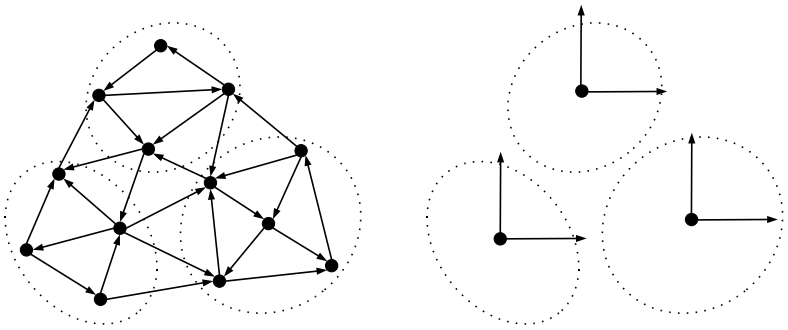

Figure 5.10: The method of Bochev et al. uses nodal aggregates (left) to determine the support of coarse edge-element basis functions. In this two-dimensional example, each nodal aggregate gives rise to two coarse-level degrees of freedom (right).

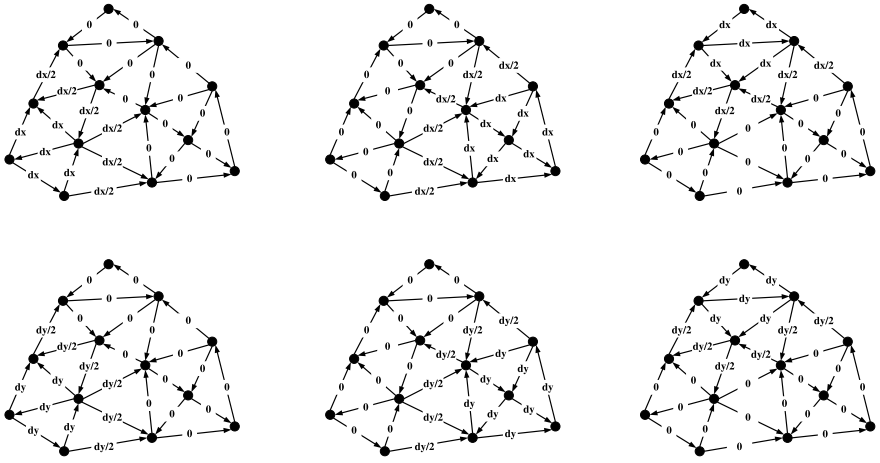

Figure 5.11: The method of Bochev et al. restricts discrete 1-form basis functions dx and dy over each nodal aggregate. Edges that straddle two nodal aggregates interpolate from both coarse DoFs with weight 1/2.

System	Grid	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_0^T\mathbb{D}_0$	250^2	63,001	0.405	11.027	1.081	3
	500^2	251,001	0.397	10.770	1.082	3
	1000^2	1002,001	0.416	11.362	1.083	4
$\mathbb{D}_1^T\mathbb{D}_1$	250^2	125,500	0.192	7.281	1.304	3
	500^2	501,000	0.180	7.030	1.307	4
	1000^2	2,002,000	0.193	7.325	1.310	4
$\mathbb{D}_0\mathbb{D}_0^T$	250^2	125,500	0.241	8.437	1.305	3
	500^2	501,000	0.331	10.901	1.308	4
	1000^2	2,002,000	0.329	10.867	1.310	4
$\mathbb{D}_1\mathbb{D}_1^T$	250^2	62,500	0.387	10.487	1.080	3
	500^2	250,000	0.399	10.858	1.082	3
	1000^2	1,000,000	0.446	12.361	1.082	4
$\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$	250^2	63,001	0.319	8.552	1.060	3
	500^2	251,001	0.339	9.028	1.060	3
	1000^2	1,002,001	0.331	8.827	1.061	4
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	250^2	125,500	0.178	6.951	1.304	3
	500^2	501,000	0.180	7.026	1.309	4
	1000^2	2,002,000	0.183	7.103	1.311	4

Table 5.1: Scaling results of the kSA method on regular quadrilateral meshes using 30 nodes per aggregate.

5.4 Numerical Results

We have applied the proposed method to the structured and unstructured meshes first examined in the context of the cSA method. The following numerical results use the testing methodology introduced in Section 4.10 with two minor changes. First, in place of preconditioned conjugate gradient iteration, we use the MINRES algorithm [35] with preconditioning. Second, an energy minimization prolongation smoother [32] is applied to the tentative prolongator. A second degree smoother is used on the first level of the multigrid hierarchy, while a standard first degree smoother is used on subsequent levels. The cost of MINRES is comparable to that of the conjugate gradient method and, in our experience, more numerically stable on the semidefinite problems considered. The energy minimization approach smooths the tentative coarse basis functions while ensuring that the near-nullspace candidates lie within the range of interpolation. Since cSA does not employ a set of candidate vectors, it is not possible to apply the energy minimization in that case. As before, a dense pseudoinverse is used to compute the coarse-level solution.

Table 5.1 reports performance figures for a series of regular quadrilateral meshes. Although inferior to cSA on a basis of work per digit of accuracy (cf. Table 4.1), kSA exhibits scalability with low operator complexity. In all cases, Lloyd aggregation with an average of 30 nodes per aggregate has been used on the first level of aggregation.

Table 5.2 reports kSA performance figures for a series of regular hexahe-

System	Grid	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_0^T\mathbb{D}_0$	25^3	17,576	0.357	9.310	1.040	2
	50^3	132,651	0.342	8.998	1.049	3
	100^3	1,030,301	0.344	9.098	1.054	4
$\mathbb{D}_1^T\mathbb{D}_1$	25^3	50,700	0.129	5.906	1.312	3
	50^3	390,150	0.146	6.583	1.374	3
	100^3	3,060,300	0.151	6.861	1.406	4
$\mathbb{D}_2^T\mathbb{D}_2$	25^3	48,750	0.085	5.107	1.364	3
	50^3	382,500	0.089	5.476	1.438	3
	100^3	3,030,000	0.091	5.707	1.478	4
$\mathbb{D}_0\mathbb{D}_0^T$	25^3	50,700	0.173	7.209	1.373	3
	50^3	390,150	0.184	7.682	1.445	3
	100^3	3,060,300	0.188	8.168	1.480	4
$\mathbb{D}_1\mathbb{D}_1^T$	25^3	48,750	0.172	6.895	1.317	3
	50^3	382,500	0.201	7.890	1.374	3
	100^3	3,030,000	0.215	8.419	1.407	4
$\mathbb{D}_2\mathbb{D}_2^T$	25^3	15,625	0.269	7.289	1.040	2
	50^3	125,000	0.315	8.368	1.049	3
	100^3	1,000,000	0.322	8.577	1.054	4
$\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$	25^3	17,576	0.165	5.243	1.023	2
	50^3	132,651	0.193	5.777	1.033	3
	100^3	1,030,301	0.176	5.528	1.038	3
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	25^3	50,700	0.108	5.150	1.246	3
	50^3	390,150	0.112	5.553	1.320	3
	100^3	3,060,300	0.113	5.727	1.357	3
$\mathbb{D}_2^T\mathbb{M}_3\mathbb{D}_2$	25^3	48,750	0.087	5.146	1.364	3
	50^3	382,500	0.092	5.581	1.442	3
	100^3	3,030,000	0.099	5.900	1.481	4

Table 5.2: Scaling results of the kSA method on regular hexahedral meshes using 100 nodes per aggregate on the finest level.

System	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_0^T\mathbb{D}_0$	84,280	0.361	9.187	1.016	3
$\mathbb{D}_1^T\mathbb{D}_1$	554,213	0.293	9.015	1.200	3
$\mathbb{D}_2^T\mathbb{D}_2$	920,168	0.194	6.909	1.231	3
$\mathbb{D}_0\mathbb{D}_0^T$	554,213	0.292	11.257	1.504	3
$\mathbb{D}_1\mathbb{D}_1^T$	920,168	0.497	18.492	1.403	3
$\mathbb{D}_2\mathbb{D}_2^T$	450,235	0.426	11.313	1.048	3
$\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$	84,280	0.411	10.530	1.016	3
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	554,213	0.337	10.313	1.219	3
$\mathbb{D}_2^T\mathbb{M}_3\mathbb{D}_2$	920,168	0.190	6.840	1.233	3

Table 5.3: Performance of kSA on the tetrahedral rocket mesh using 100 nodes per aggregate.

System	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_0^T\mathbb{D}_0$	91,578	0.322	8.374	1.029	3
$\mathbb{D}_1^T\mathbb{D}_1$	585,543	0.253	8.894	1.327	3
$\mathbb{D}_2^T\mathbb{D}_2$	949,646	0.163	6.887	1.354	4
$\mathbb{D}_0\mathbb{D}_0^T$	585,543	0.281	11.101	1.528	3
$\mathbb{D}_1\mathbb{D}_1^T$	949,646	0.258	9.270	1.363	3
$\mathbb{D}_2\mathbb{D}_2^T$	455,683	0.424	11.202	1.042	3
$\mathbb{D}_0^T\mathbb{M}_1\mathbb{D}_0$	91,578	0.445	11.733	1.029	3
$\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$	585,543	0.348	12.098	1.383	3
$\mathbb{D}_2^T\mathbb{M}_3\mathbb{D}_2$	949,646	0.171	7.054	1.351	4

Table 5.4: Performance of kSA on a three-holed mesh using 100 nodes per aggregate.

dral meshes using an average of 100 nodes per aggregate on the first level of aggregation. Like the two-dimensional case, kSA scales well while maintaining low operator complexity. In several cases, such as the 1-form problem $\mathbb{D}_1^T\mathbb{D}_1$ and 2-form problem $\mathbb{D}_2^T\mathbb{M}_3\mathbb{D}_2$, kSA requires considerably less work per digit of accuracy.

Recall the unstructured tetrahedral mesh illustrated in Figure 4.4. Performance of the cSA method on this example (cf. Table 4.3) is noticeably worse than the corresponding problems on structured meshes. Furthermore, when an innerproduct is introduced to the 1-form problem (i.e. $\mathbb{D}_1^T\mathbb{M}_2\mathbb{D}_1$) cSA performance substantially degrades. In contrast, kSA performance, reported in Table 5.3, is insensitive to the existence of an innerproduct. Ignoring the scalar problems, which are better handled by standard smoothed aggregation, kSA performance is generally better than that of cSA.

Another unstructured tetrahedral mesh with three holes is illustrated in Figure 5.12. As shown by the performance figures in Table 5.4, solver performance does not substaintally degrade when an innerproduct is introduced on this problem.

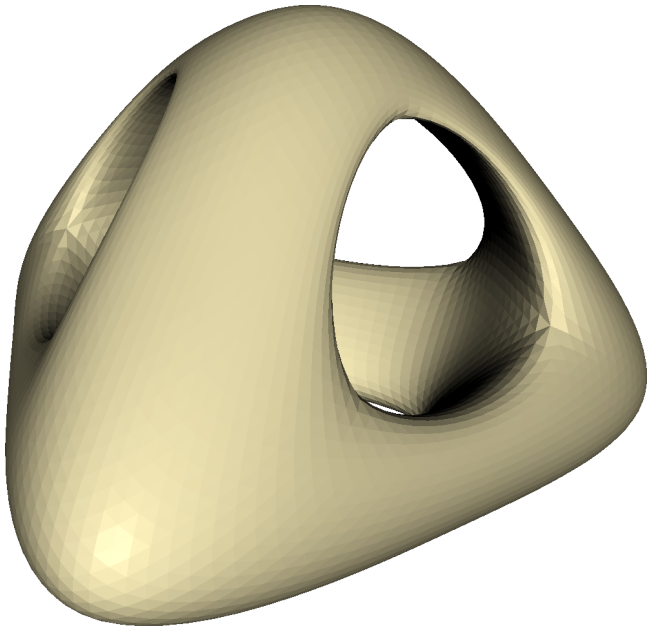

Figure 5.12: Tetrahedral mesh with three holes.

5.5 Coordinate Systems

As mentioned in Section 5.1.2, some manifolds are representable with multiple coordinate systems. The torus, for instance, is naturally embedded in three-dimensional Euclidean space with coordinates (x, y, z). This choice of coordinates gives rise to the 1-form basis $\{dx, dy, dz\}$, which we refer to as the *embedded basis*, is illustrated in Figure 5.13. An alternative set of coordinates (u, v), where $u, v \in [0, 2\pi)$ also parameterizes the surface[2]. Figure 5.14 depicts the associated 1-form basis $\{du, dv\}$, which we refer to as the *intrinsic basis*.

Given that both bases satisfy the expectations of the proposed method, a natural question arises: which basis is preferable? As Table 5.5 reveals, the choice of basis does not significantly change the solver's rate of convergence. Since the (u, v) 1-form basis consists of only two vectors while the (x, y, z) basis contains three, the operator complexity of the latter basis is higher. Since the methods' convergence is comparable, the (u, v) basis, with its lower complexity, requires less work per digit of accuracy.

The similarity between convergence rates is attributable to the fact that, over each aggregate, the embedding basis closely approximates the intrinsic basis. Indeed, by analogy to the *inverse function theorem* of smooth manifolds, the tangent plane at that point on the torus is, locally, a suitable coordinate system for the surface. As a result, over each aggregate, the subspace of the em-

[2]We ignore the fact that the (u, v) coordinates are only local coordinates. In reality, an atlas of four charts, or mappings from a two-dimensional disc to the surface of the torus, is needed to smoothly parameterize the surface.

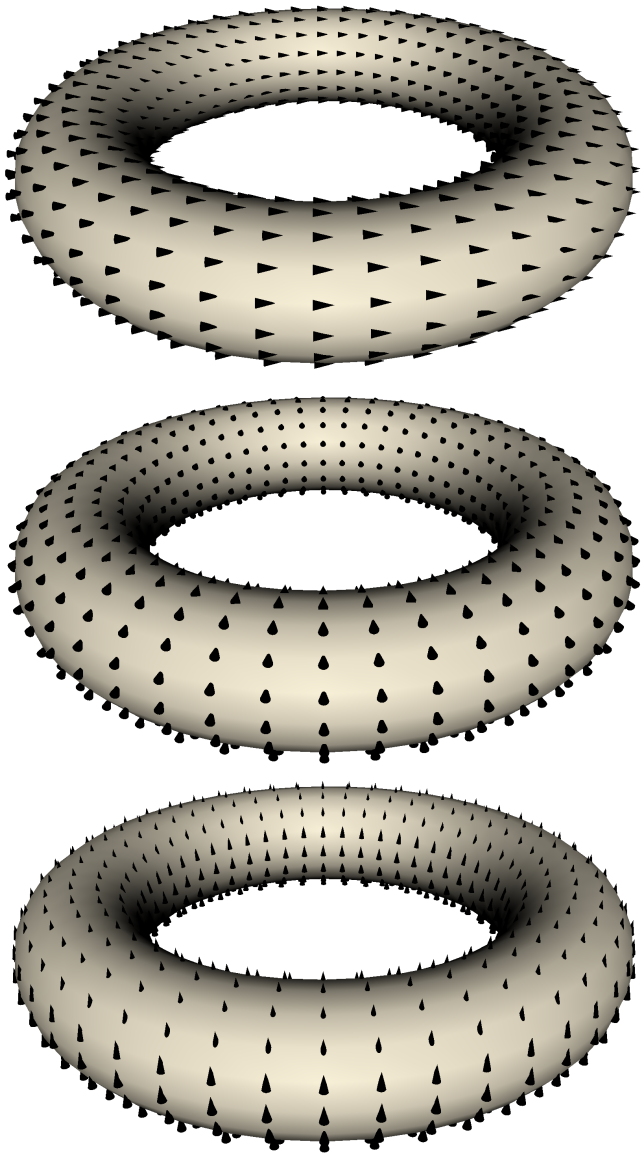

Figure 5.13: The 1-form basis elements dx, dy, and dz, corresponding to the Euclidean coordinates (x, y, z).

System	Basis	Unknowns	Convergence	WPD	OC	Levels
$\mathbb{D}_1\mathbb{D}_1^T$	dx, dy, dz	36,864	0.220	8.060	1.324	3
$\mathbb{D}_1\mathbb{D}_1^T$	du, dv	36,864	0.214	10.351	1.732	3
$\mathbb{D}_1\mathbb{M}_2\mathbb{D}_1^T$	dx, dy, dz	36,864	0.227	8.250	1.652	3
$\mathbb{D}_1\mathbb{M}_2\mathbb{D}_1^T$	du, dv	36,864	0.209	10.204	1.733	3

Table 5.5: Comparison of kSA performance using different 1-form bases for the torus mesh.

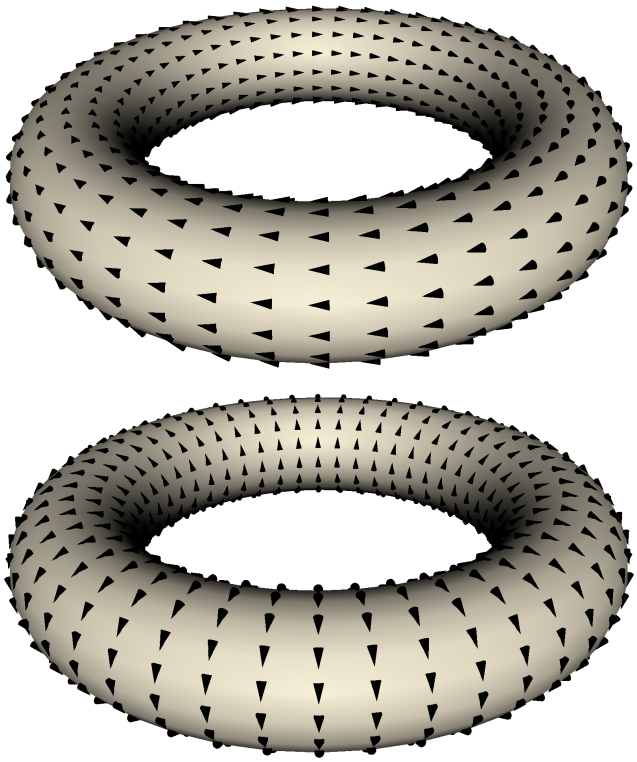

Figure 5.14: The 1-form basis elements du and dv, corresponding to the intrinsic coordinates (u, v).

bedded basis that coincides with the direction of the aggregate's surface normal is unnecessary.

We verify the previous claims with the following numerical experiment: the candidate matrices B_{uv} and B_{xyz} are used to construct tentative prolongators T_{uv} and T_{xyz} using the same aggregation of nodes in both cases. The error of the best 2-norm approximation of the columns of T_{uv} by $\mathcal{R}(T_{xyz})$ is given by the expression

$$E_{uv} = (I - T_{xyz}(T_{xyz}^T T_{xyz})^{-1} T_{xyz}^T) T_{uv}. \tag{5.3}$$

Relative errors are determined by computing the 2-norm of each column of E_{uv} and dividing by the norm of the corresponding column of T_{uv}. Figure 5.15 illustrates the distribution of relative approximation errors across two discretizations of the torus. In both cases the average approximation error is small, only 0.51% in the coarser discretization and 0.13% in the finer mesh.

Although solver performance using the intrinsic basis is superior to that of the embedded basis, the embedded basis has several advantages. While the intrinsic coordinates of a torus are evident, complex meshes do not admit such simple mappings. In contrast, a mesh embedding is often available. Furthermore, in most cases of interest, the dimension of the mesh embedding is not significantly larger than the dimension of the manifold itself, and therefore the redundancy of the embedded basis is limited.

5.6 Combinatorial Laplacians

Given a set of boundary operators $\partial_0, \partial_1, \ldots, \partial_N$, the k-th combinatorial Laplacian [23, 33] is defined as

$$\Delta_k = \partial_k^T \partial_k + \partial_{k+1} \partial_{k+1}^T. \tag{5.4}$$

Since the discrete derivative is defined to be the adjoint of the boundary operator (Section 3.1),

$$\Delta_k = \mathbb{D}_{k-1} \mathbb{D}_{k-1}^T + \mathbb{D}_k^T \mathbb{D}_k \tag{5.5}$$

is an equivalent definition. Chapter 8 discusses how the topological information exposed by combinatorial Laplacians is used to determine the coverage of sensor networks. So far, we have considered the systems $\mathbb{D}_k^T \mathbb{D}_k$ and $\mathbb{D}_k \mathbb{D}_k^T$ which are related to the continuous operators $\boldsymbol{\delta d}$ and $\boldsymbol{d\delta}$ respectively. The combinatorial Laplacian is related to the Laplace-de Rham, or Hodge Laplacian $\boldsymbol{\delta d} + \boldsymbol{d\delta}$.

When solving problems involving combinatorial Laplacians, we apply an adaptive version of the kSA method. This choice is due, in part, to the fact that the approach to the sensor network coverage problem discussed in Chapter 8 is coordinate-free. Furthermore, even when coordinates are available and the kSA method is applied, we observe problem size-dependent convergence rates, indicating that the k-form basis does not fully capture slow-to-converge modes.

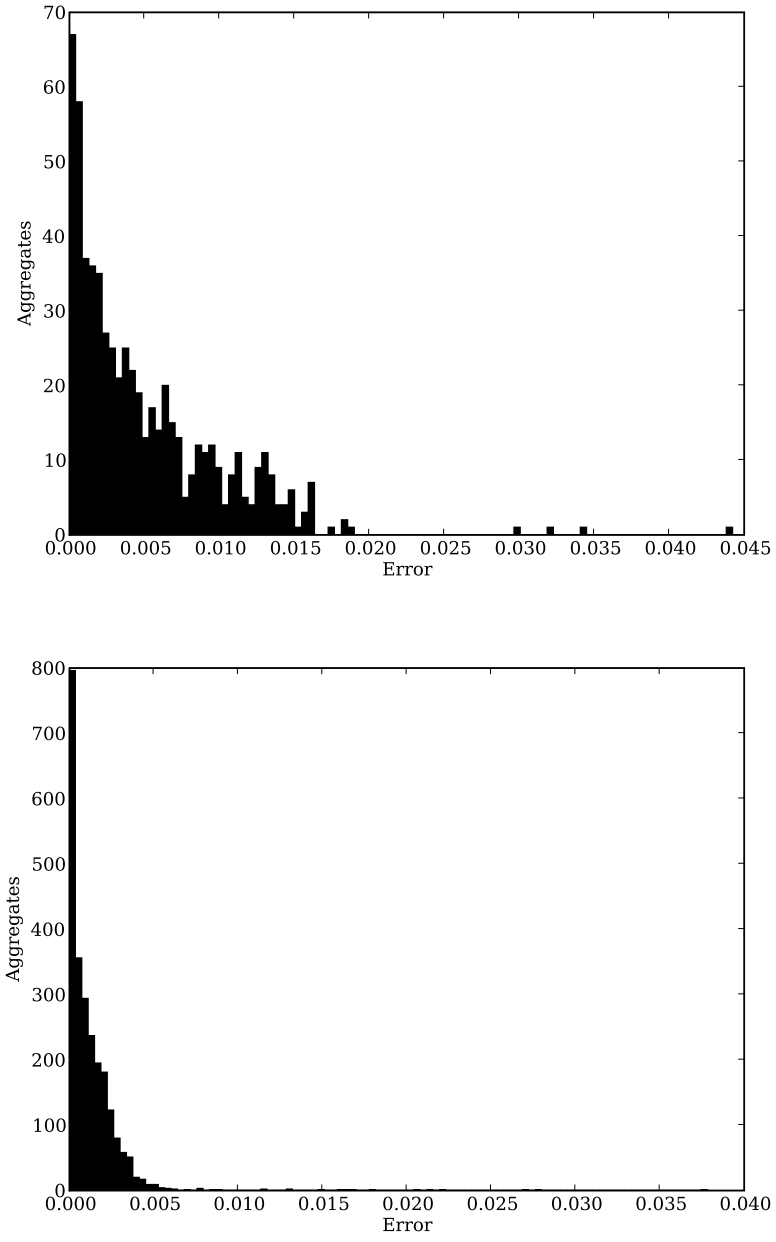

Figure 5.15: Distribution of per-aggregate approximation errors. For each aggregate, a localized basis with elements $\{dx, dy, dz\}$ is used to approximate the basis $\{du, dv\}$. The reported error is a measure of the difference between the vectors du and dv and the best approximation to du and dv using the three-element basis. The top figure reflects the distribution of errors for a torus mesh with 6,144 triangles while the mesh for the bottom figure has 24,576. As the mesh is refined, the approximation errors are reduced.

System	Unknowns	Convergence	WPD	OC	Candidates	Levels
Δ_1	554,213	0.505	22.293	1.652	6	3
Δ_2	920,168	0.491	22.796	1.762	6	3

Table 5.6: Adaptive kSA performance on the combinatorial Laplacians of the unstructured tetrahedral rocket mesh.

This observation is supported by the results of [8].

Coarsening in adaptive kSA consists of one level of Lloyd aggregation followed by standard SA aggregation [44] on subsequent levels. Using this aggregation of variables, the adaptive SA algorithm computes a set of near-nullspace candidates as explained in Section 2.1.1. Although our primary interest is in the combinatorial Laplacians that arise in sensor network problems, we report performance figures for the operators Δ_1 and Δ_2 of the unstructured rocket mesh in Table 5.6. Here, six near-nullspace candidates have been generated by the adaptive setup algorithm. First level aggregates, computed with Lloyd aggregation, consist of an average of 200 nodes each. A second degree prolongation smoother has been used on the first level.

Chapter 6

Lloyd Aggregation

Recall the hierarchy construction algorithm of algebraic multigrid based on smoothed aggregation (SA) discussed in Section 2.1. Aggregates computed during the aggregation phase of SA determine the support region of coarse basis functions in the tentative prolongator. Since the tentative prolongator in turn determines the smoothed prolongator and Galerkin product $P^T AP$, the selection of aggregates strongly influences the efficiency and cost of the resultant multigrid cycle.

In this Chapter we describe an aggregation method based on Lloyd's method [30]. We show that the proposed method, which we call *Lloyd Aggregation*, is an appropriate and effective method for discrete k-form problems. Furthermore, the additional flexibility offered by Lloyd aggregation supports a time-memory tradeoff between work per digit of accuracy (WPD) and operator complexity.

6.1 Standard Aggregation

As a basis for comparison, we consider the aggregation algorithm introduced by Vaněk et al. [44], which we refer to as *standard aggregation*. Standard aggregation favors small 1-ring neighborhoods, such as those illustrated by Figure 6.1, when constructing aggregates. Aggregates of this form are well-suited to problems discretized with nodal finite elements.

Algorithm 6.1 details the steps of the standard aggregation algorithm. The method takes a strength of connection matrix S as input and produces a number of aggregate sets C_i. The strength of connection matrix is typically obtained by dropping weak connections from A, and therefore consists of a subset of its nonzero entries.

The algorithm proceeds in three serial passes, the first of which identifies nodes whose entire 1-ring neighborhood is available for aggregation, i.e. a subset of U. A second pass attempts to place unaggregated nodes into any neighboring aggregate created in the first pass, while the final pass creates aggregates from the few remaining nodes without concern for their structure.

Since Algorithm 6.1 is a greedy serial algorithm, it is sensitive to the ordering of nodes. As Figure 6.2 demonstrates, a lexicographical ordering of the nodes of a regular quadrilateral mesh produces square aggregates while a ran-

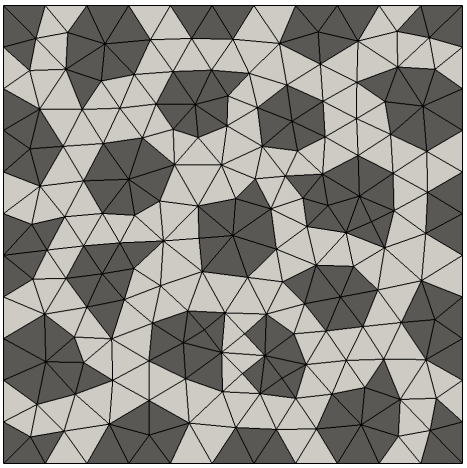

Figure 6.1: The standard aggregation algorithm typically aggregates nodes and their 1-ring neighbors.

Algorithm 6.1: `standard_aggregation`(S)

```
1  N ⇐ #rows(S)
2  U ⇐ {0, 1, ... N − 1}
3  P ⇐ ∅
4  for i = 0, 1, ..., N − 1
5      if i ∈ U and Neighbors(S, i) ⊂ U
6          C_i ⇐ Neighbors(S, i) ∪ {i}
7          U ⇐ U \ C_i
8          P ⇐ P ∪ C_i
9      end
10 end
11 for i = 0, 1, ..., N − 1
12     if i ∈ U and Neighbors(S, i) ∩ P ≠ ∅
13         k ⇐ min{k : j ∈ C_k}
14         C_k ⇐ C_k ∪ {i}
15         U ⇐ U \ {i}
16     end
17 end
18 for i = 0, 1, ..., N − 1
19     if i ∈ U
20         C_i ⇐ (Neighbors(S, i) ∪ {i}) ∩ U
21         U ⇐ U \ C_i
22     end
23 end
```

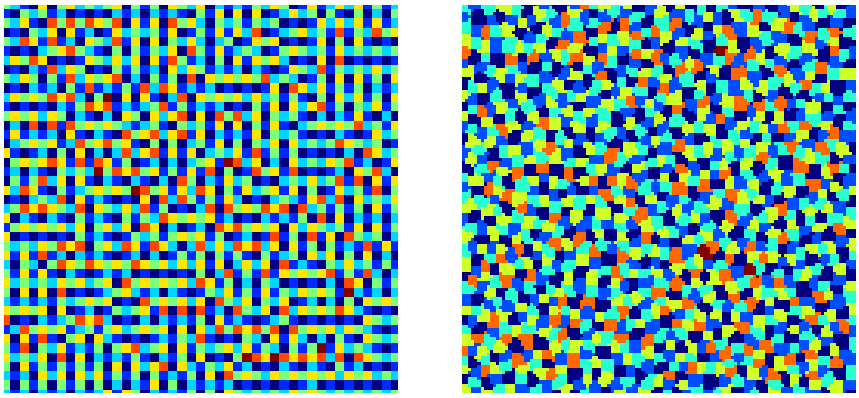

Figure 6.2: The standard aggregation method is sensitive to the ordering of unknowns. Lexicographical ordering (left) results in square aggregates while a random ordering (right) yields irregular aggregates on the two-dimensional Poisson problem discretized with Q1 finite elements.

dom ordering does not. This difference is not merely aesthetic, since the cost and efficiency of the resulting multigrid cycle depends on the structure of the aggregates. In this particular comparison, the two orderings give rise to hierarchies with comparable operator complexities (memory cost). However the WPD of the multigrid cycle in the randomized case is 49.6% greater than that of the lexicographical ordering, indicating inferior interpolation accuracy. Given the standard method's sensitivity to node ordering, a quality that cannot be assumed in a general setting, the numerical results reported in this chapter are for randomized orderings of the degrees of freedom.

Remark 1. *There are numerous possible variants of Algorithm 6.1. For instance, during the second pass of Algorithm 6.1 the unaggregated node is placed in the neighboring aggregate with the fewest members, in an attempt to moderate aggregate size. Alternatively, a parallel analog of Algorithm 6.1 is realized by computing a distance-2 maximal independent set of S in parallel [31, 2], and then aggregating nodes with their nearest set member.*

6.2 Lloyd's Method

The proposed method, Lloyd aggregation, is a natural extension of Lloyd's algorithm [30] to graphs. Therefore, it is instructive to first present Lloyd's method in a more conventional setting. Consider the problem of distributing N points, or *centers*, in a manner that minimizes the average distance of an arbitrary point from a given domain to the nearest center. Practical examples of this problem include quantization of image data, the placement of resources such as hospitals, and the selection of quadrature nodes [19]. Lloyd's method is an iterative algorithm for (approximately) solving such problems.

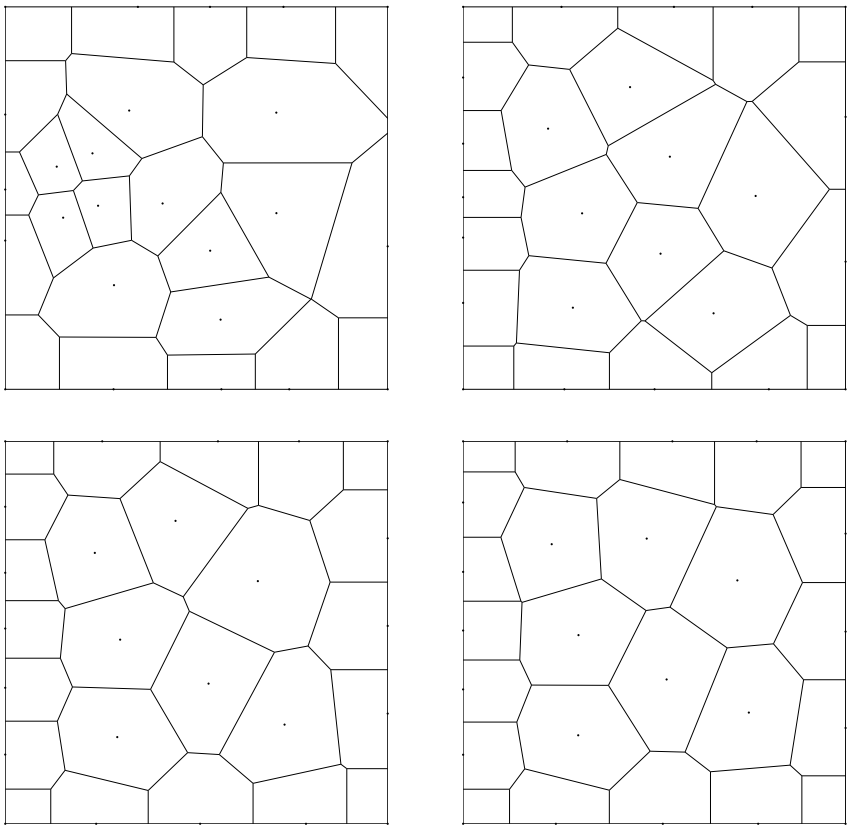

Figure 6.3: Lloyd's algorithm applied to 25 points in the plane. Shown are the centers after 1 iteration (upper left), 5 iterations (upper right), 10 iterations (lower left), and 15 iterations (lower right). At each iteration, each center is moved to the centroid of its Voronoi region.

Beginning with a given set of centers — e.g. randomly chosen or uniformly spaced — each iteration of the algorithm consists of two steps. In the first step, the *Voronoi* region of each center, i.e. the set of points that are closer to a given center than any other center, is computed. In the second step, the centers are moved to the *centroid* of their Voronoi regions. While the notions of domain, points, and centroid are problem (and metric) dependent, the procedure is widely applicable. Figure 6.3 illustrates the problem of selecting 25 centers from the unit square such that the average Euclidean distance from a point within the square to its nearest center is minimized. In this case, the centroid corresponds to the center of mass of each Voronoi region. Despite starting with a random set of centers, the algorithm generally converges to a reasonable, but not necessarily optimal, set of centers. For further detail, we refer the interested reader to the work of Du et al. [19].

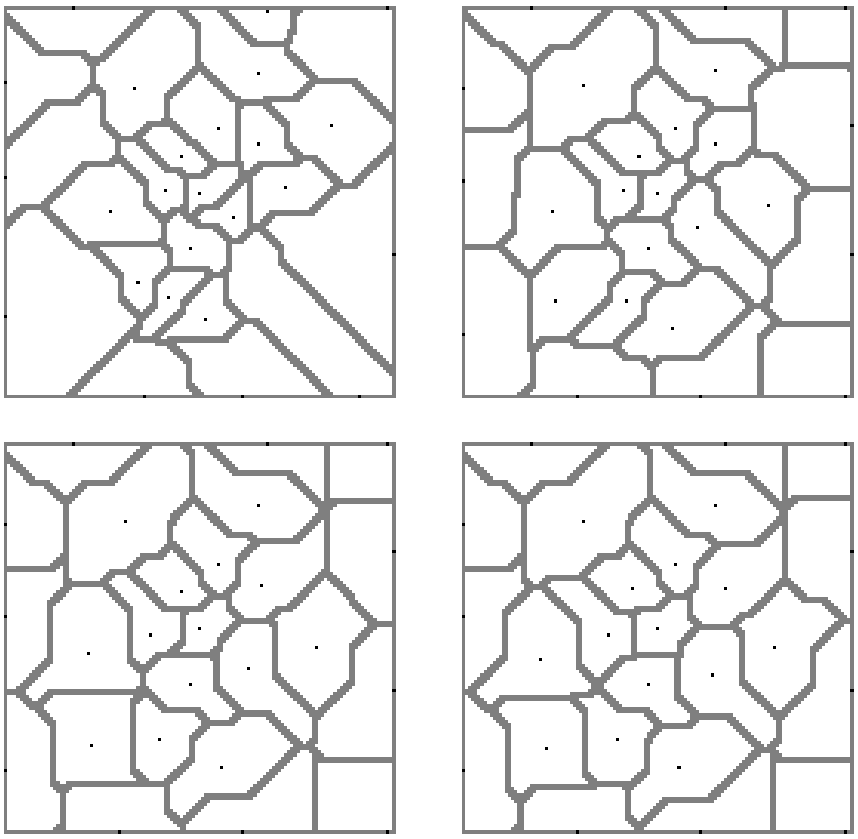

Figure 6.4: Sequence of four iterations of Lloyd aggregation on a regular quadrilateral mesh.

6.3 Proposed Method

Lloyd aggregation is a direct application of Lloyd's algorithm to the matrix graph of S, the same strength of connection matrix used in standard aggregation. In this case, the domain consists of the graph nodes of S, and a random subset of N nodes serve as the initial centers. The Voronoi regions are determined by computing the nearest center to each graph node. The distance between two nodes is simply the length of the shortest path through the graph which joins them. For now, we assume that each graph edge has unit distance. A centroid of a region is any node that is farthest from the boundary of that region. When a region has multiple centroids, the new center is chosen arbitrarily among them. Figure 6.4 demonstrates this procedure with 25 centers on a regular discretization of the unit square.

6.3.1 Implementation

There are multiple ways to implement the aforementioned steps of Lloyd aggregation. One approach is to use an efficient multi-source version of Dijkstra's

Algorithm 6.2: `modified_bellman_ford`(S,Centers)

```
1  Distance ⇐ [∞, ∞, ..., ∞]
2  NearestCenter ⇐ [−1, −1, ..., −1]
3  for c ∈ Centers
4      Distance[c] ⇐ 0
5      NearestCenter[c] ⇐ c
6  end
7  while True
8      Finished ⇐ True
9      for (i, j) ∈ Nonzeros(S)
10         d_ij ⇐ S[i, j]
11         if Distance[i] + d_ij < Distance[j]
12             Distance[j] ⇐ Distance[i] + d_ij
13             NearestCenter[j] ⇐ NearestCenter[i]
14             Finished ⇐ False
15         end
16     end
17     if Finished
18         return Distance, NearestCenter
19     end
20 end
```

shortest path algorithm to compute the nearest center to each node in the graph. However, for our domain of interest, a modified form of the Bellman-Ford algorithm[15] is a more appropriate choice. The modified Bellman-Ford procedure, detailed in Algorithm 6.2, takes as input the strength of connection matrix S and a set of centers. For now, we assume that all nonzeros of S have the value 1, corresponding to unit distances on all graph edges. The modified Bellman-Ford algorithm propagates not only the distance to the nearest center, but also the index of that center. Upon completion, the vector `NearestCenter` encodes the Voronoi regions surrounding each center and `Distance` the distance to the nearest center.

Algorithm 6.2 terminates as soon as the distance vector is correct. Specifically, when the distance vector remains unchanged during an iteration — i.e. a fixed-point has been reached — then the distances, and Voronoi regions, are correct. In the context of Lloyd aggregation, where the number of centers is a fixed ratio of the total number of nodes, the distance to the nearest center is small. Furthermore, the k-th iteration of Bellman-Form ensures that all nodes within a path of length k from a center are correctly determined. Therefore, only a small number of iterations are needed to compute the nearest center to each node. Furthermore, Algorithm 6.2 is inherently parallel. Indeed, each iteration of the method resembles a sparse matrix-vector product, with the expression $y(i) = \min(y(i), A(i,j) + x(i))$ taking the place of $y(i) = y(i) + A(i,j) * x(i)$. Given the small number of required iterations and the amenability of the algorithm to parallel implementation, Algorithm 6.2 is well-suited to Lloyd aggre-

Algorithm 6.3: `lloyd_aggregation`(S,Centers)

```
for iter = 1,...,Iterations
    Distance,NearestCenter ⇐ modified_bellman_ford(S,
        Centers)
    Border ⇐ ∅
    for (i,j) ∈ Nonzeros(S)
        if NearestCenter[i] ≠ NearestCenter[j]
            Border ⇐ Border ∪ {i,j}
        end
    end
    Distance,x ⇐ modified_bellman_ford(S,Border)
    Centers ⇐ { i : Distance[i] > Distance[j] ∀
        NearestCenter[i] == NearestCenter[j]}
end
return extract_aggregates(NearestCenter)
```

gation.

When computing the Voronoi regions with modified Bellman-Ford, the distance of the centers are initialized to 0 and the other nodes with the value ∞. Upon completion, each node has determined the index of the nearest center and its distance from that center. As shown in Algorithm 6.3, this process is also used to compute a centroid of each aggregate by initializing all boundary nodes, i.e. those nodes that have a neighbor in a different aggregate, with the value 0 and all remaining nodes with ∞. Once distances from the boundary nodes have been computed, any node with maximum distance is selected as the aggregate centroid. In a parallel setting, this information is propagated across an aggregate in an iterative fashion, thus making all aspects of Lloyd aggregation highly parallel.

6.4 Results

In this section we compare the proposed method to the standard aggregation of Vaněk et al. [44]. Of principle interest is the *operator complexity* of the multigrid hierarchy and the *work per digit* of accuracy (WPD) of the multigrid cycle. Operator complexity is a rough metric for the cost of constructing and storing the multigrid hierarchy, while WPD measures the computational efficiency of the solver. Although the effect of operator complexity is implicit in WPD, it is important to consider both measures together. Among methods with similar cycle efficiencies, the method with the smallest memory and setup cost is generally preferred. Furthermore, methods that exceed the memory capacity of a given platform are impractical, independent of their WPD. As a result, it is worthwhile to consider not only methods with optimal WPD, but also Pareto optimal methods, i.e. methods with the best WPD for a given operator complexity limit. By varying the number of centers used in the aggregation phase,

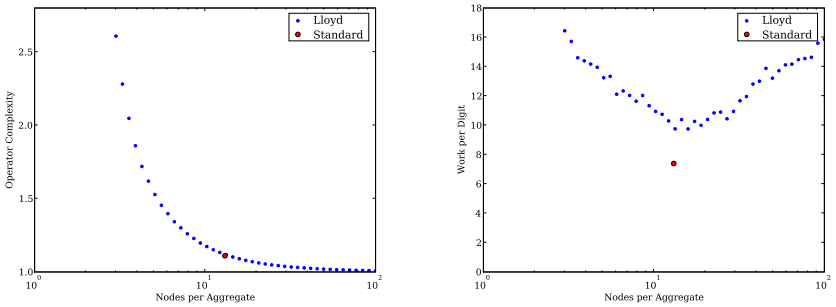

Figure 6.5: Comparison of aggregation methods on a regular quadrilateral mesh with isotropic diffusion.

we explore the time-memory tradeoff facilitated by Lloyd aggregation.

6.4.1 Methodology

The following results compare Lloyd aggregation to standard aggregation in the context of algebraic multigrid based on smoothed aggregation. In every test, a multigrid hierarchy is constructed with one of the two aggregation methods and is used to precondition conjugate gradient iteration with a V(1,1) cycle using a symmetric Gauss-Seidel sweep during pre- and post-smoothing. We use an energy minimization approach [32] for prolongator smoothing. Coarsening proceeds until the number of unknowns on the coarsest grid falls below 100. Beginning with a random right hand side, the system $Ax = b$ is iterated upon until the norm of the residual $||b - Ax||$ is reduced by 10 orders of magnitude. Unless stated otherwise, we apply the same aggregation method, and coarsening ratio in the case of Lloyd aggregation, on all levels of the multigrid hierarchy. The 'Nodes per Aggregate' figures represent the average number of degrees of freedom per aggregate in the *first level* of aggregation only.

6.4.2 Isotropic Diffusion

As our first example, we consider isotropic diffusion on a regular quadrilateral mesh with 128^2 elements. Standard Q1 finite elements are used to discretize $-\Delta u = f$ in weak form. Dirichlet boundary conditions are imposed at all boundary nodes.

The numerical results reported in Figure 6.5 indicate that this model problem is readily solved by both methods. Here, the standard method achieves a WPD of 7.39 while maintaining low operator complexity. The best Lloyd aggregation results have comparable operator complexity, but exhibit inferior WPD figures. Given the modest cost of the standard approach in this particular example, the ability of Lloyd aggregation to produce more economical hierarchies is of marginal benefit. Considering that this is an ideal problem for

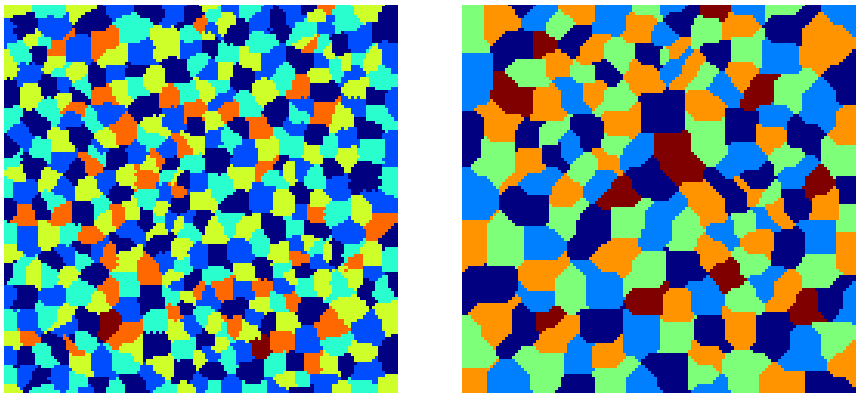

Figure 6.6: Lloyd aggregates with approximately 40 (left) and 100 (right) nodes per aggregate.

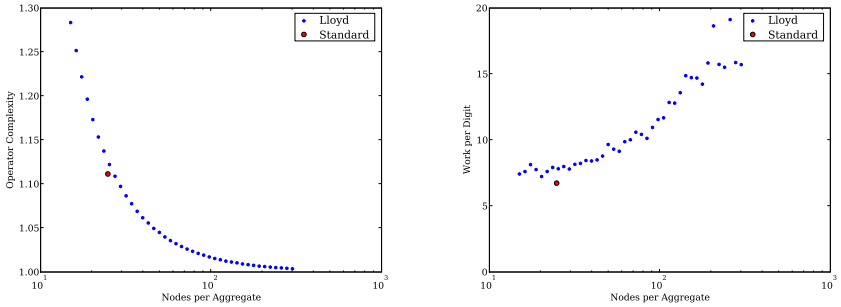

Figure 6.7: Comparison of aggregation methods on an unstructured tetrahedral rocket mesh with isotropic diffusion.

the standard algorithm, the performance of the Lloyd approach is acceptable. Figure 6.6 illustrates sample Lloyd aggregates with 40 and 100 nodes per aggregate. Despite the increase in nodes per aggregate, the cycle efficiency of the Lloyd-based method does not substantially degrade.

As shown in Figure 6.7, the performance characteristics of the previous structured example carry over to the unstructured tetrahedral rocket mesh (cf. Figure 4.4) discretized with P1 finite elements. Using an average of 30 nodes per aggregate, the Lloyd-based solver has nearly the same operator complexity as the standard method, but requires 16% more work per digit of accuracy. While Lloyd aggregation is able to coarsen more quickly than the standard method, the low cost of the standard hierarchy implies that aggressive coarsening is unnecessary within this class of problems.

6.4.3 Anisotropic Diffusion

Anisotropic diffusion is the subject of our second example, which uses the same discretization as the isotropic case in Section 6.4.2. Before the aggregation

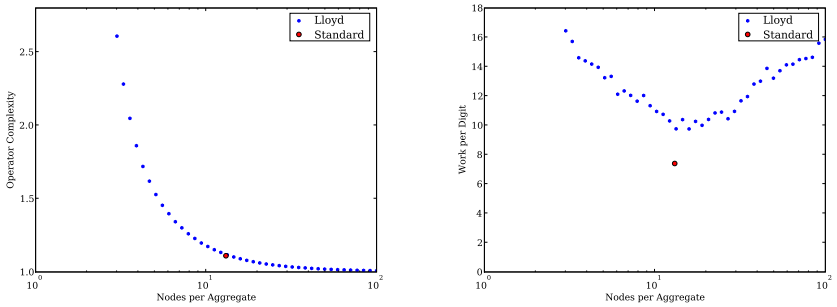

Figure 6.8: Comparison of aggregation methods on a regular quadrilateral mesh with anisotropic diffusion.

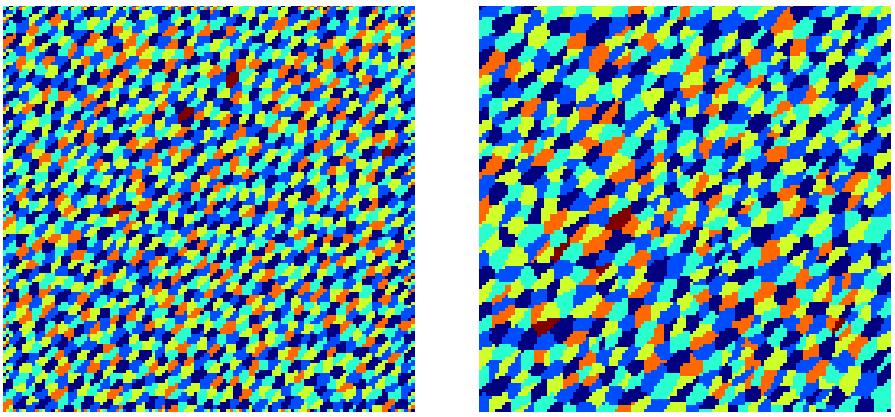

Figure 6.9: Anisotropic aggregates for standard aggregation (left) and Lloyd aggregation with approximately 20 nodes per aggregate (right).

setup we apply a robust strength of connection measure to the matrix [39] to ensure that only strongly connected neighbors are aggregated. Like the isotropic results, the standard method requires less work per digit of accuracy, while Lloyd aggregation can produce smaller hierarchies without significant performance degradation. Figure 6.9 depicts aggregates for this example.

6.4.4 Dual Meshes

The examples of Sections 6.4.3 and 6.4.3 were discretized by associating basis functions to the nodes of *primal* meshes. In this case, the sparsity pattern of A (and therefore the strength of connection matrix S) is simply the graph determined by the nodes and edges of the primal mesh. In contrast, scalar problems that reside on the *dual* mesh associate basis functions to the top-level elements, or, equivalently, the centers of the top-level elements. The sparsity structure of the dual problem is the same as the dual graph of the mesh, i.e. an edge exists between pairs of top-level elements that share a common face.

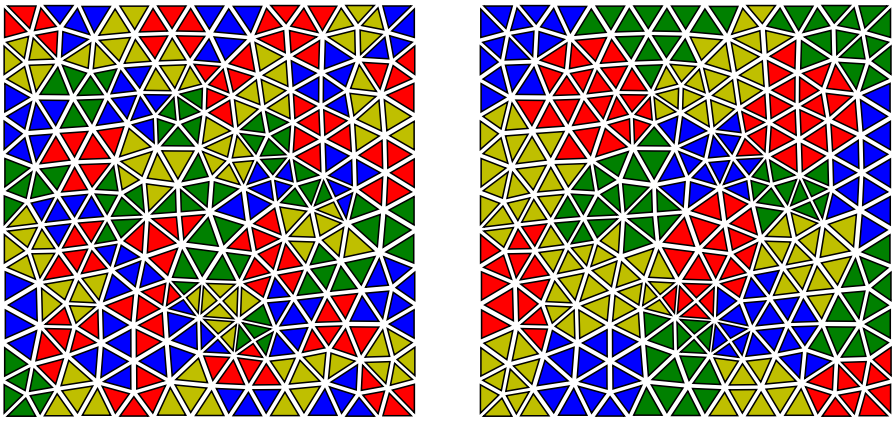

Figure 6.10: Standard aggregates (left) and Lloyd aggregates (right) with 15 nodes per aggregate on the dual mesh.

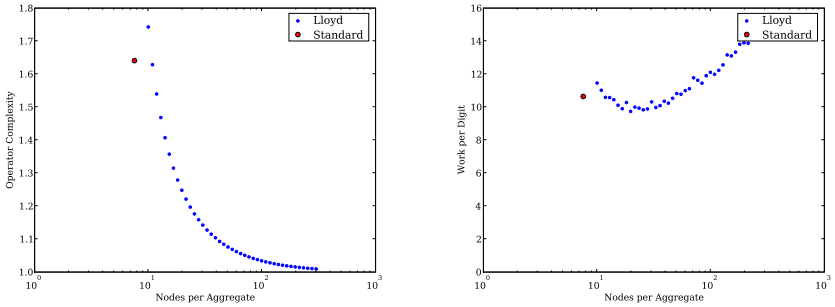

Figure 6.11: Comparison of aggregation methods on the dual of an unstructured tetrahedral rocket mesh.

Figure 6.10 illustrates dual mesh aggregates for the two methods.

The sparsity structure of the dual problem is unlike that of the primal. For instance, the primal problem $\mathbb{D}_0^T\mathbb{D}_0$ on the unstructured tetrahedral rocket mesh (cf. Figure 4.4) has an average of 14.1 nonzeros per row (13.1 adjacencies). However, since a tetrahedron has at most 4 adjacencies, the dual problem $\mathbb{D}_2\mathbb{D}_2^T$ has only 4.9 nonzeros per row (3.9 adjacencies) on average. The limited size of the 1-ring neighborhoods of the dual mesh severely undermines the selection strategy of the standard aggregation method, leading to high operator complexities.

Figure 6.11 reports performance figures for the dual system $\mathbb{D}_2\mathbb{D}_2^T$. Due to a slow coarsening rate, the standard method's operator complexity (1.64) is considerably higher than the scalar problem on the primal mesh. On the other hand, Lloyd aggregation, which is forced to coarsen at a predefined rate, yields a more economical hierarchy. Moreover, the Lloyd-based solver achieves a comparable, and sometimes better WPD, at substantially lower cost. In particular,

the Lloyd result with approximately 55 nodes per aggregate produces the same WPD as the standard method, with an operator complexity of only 1.07.

6.4.5 Discrete k-forms

Nodal discretizations on the primal mesh (e.g. $\mathbb{D}_0^T\mathbb{D}_0$) were the subject of Sections 6.4.2 and 6.4.3. Similarly, nodal discretizations on the dual mesh (e.g. $\mathbb{D}_2\mathbb{D}_2^T$ in 3D) were examined in Section 6.4.4. In this section, we consider non-nodal discretizations, such as those where the degrees of freedom are associated with mesh edges and faces.

Matrices arising in k-form discretizations, like those of mesh duals, have sparsity structures which differ substantially from those of standard (primal) nodal discretizations. This implies that the standard aggregation method performs poorly on discrete k-form problems. As shown in Figure 6.12, standard aggregates for the discrete 1-form operator $\mathbb{D}_1^T\mathbb{D}_1$ are small and irregularly shaped. Together, these properties suggest that multigrid hierarchies produced by the standard method will exhibit high operator complexities. In contrast, Lloyd aggregates for the same problem are larger in size and more naturally shaped. The average aggregate size determines the rate of coarsening and the number of levels in the multigrid hierarchy. Aggregate shape generally affects the average number of adjacent aggregates, which determines the sparsity structure of the Galerkin product P^TAP. Therefore, more irregular aggregates generally create additional fill in the hierarchy while smaller aggregates extend the length of the hierarchy.

The following numerical results use the methodology laid out in Section 6.4.1 with the exception that Lloyd aggregation is used only on the *first* level of aggregation. As discussed in Section 5.2, the coarse basis functions produced by the kSA method, which resemble vector-valued nodal discretizations on coarser meshes, are appropriate candidates for standard aggregation.

Figures 6.13 and 6.14 report performance results for the k-form problems $\mathbb{D}_1^T\mathbb{D}_1$ and $\mathbb{D}_2^T\mathbb{D}_2$ on a regular hexahedral mesh with 25^3 elements. In the 1-form case, the standard method's operator complexity of 2.51 is substantially higher than Lloyd aggregation with 50 or more nodes per aggregate. For instance, the Lloyd result with 100 nodes per aggregate has an operator complexity of 1.19 while requiring 28% less work per digit of accuracy. In the 2-form problem, the performance disparity between the aggregation methods grows even larger Compared to the standard method, Lloyd aggregation with 100 nodes per aggregate is more than three times faster, while maintaining a modest operator complexity.

As shown by Figures 6.15 and 6.16, the relative performance of the two methods carries over to unstructured meshes. Again, Lloyd aggregation with 100 or more nodes per aggregate offers a favorable balance between low operator complexity and work per digit of accuracy. Furthermore, modifying the rate of

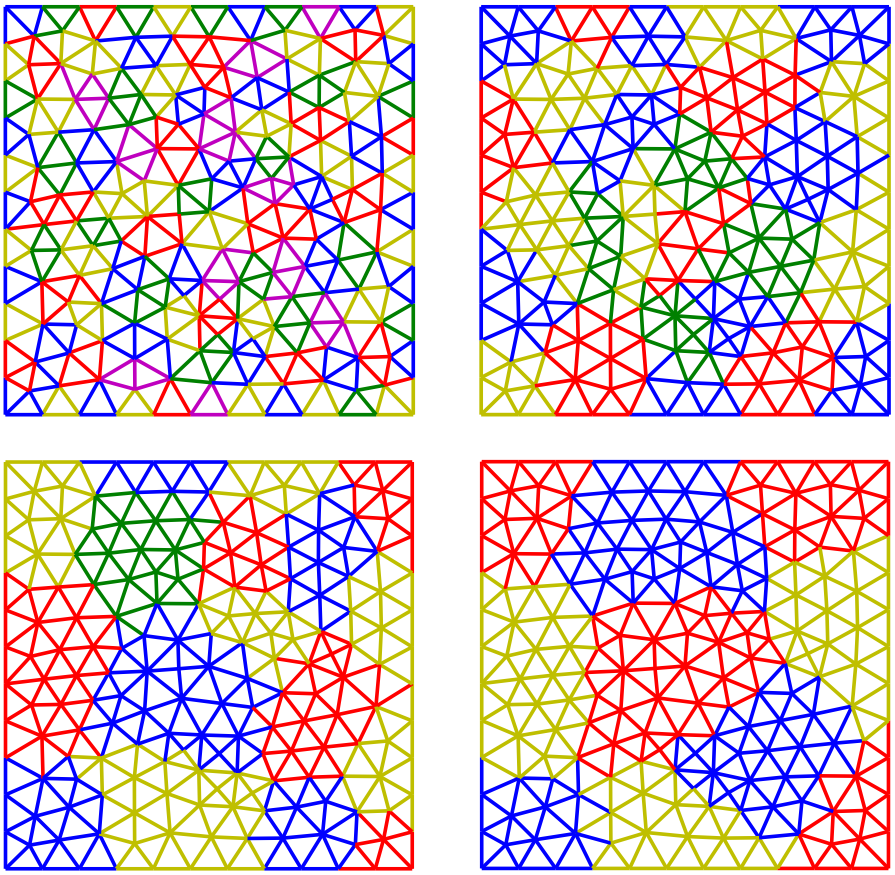

Figure 6.12: Standard aggregates (upper left) and Lloyd aggregates with an average of 20 (upper right), 30 (lower left), and 50 (lower right) nodes per aggregate for the 1-form operator $\mathbb{D}_1^T\mathbb{D}_1$.

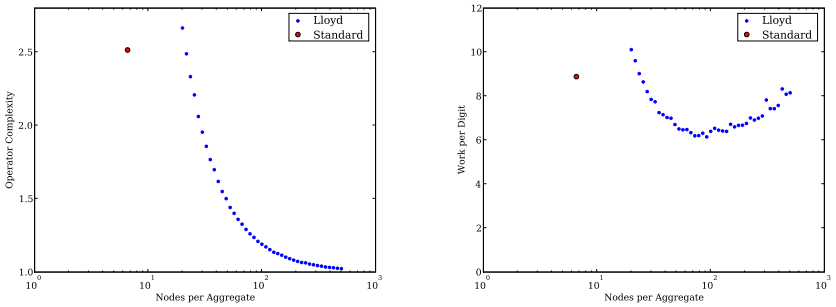

Figure 6.13: Comparison of aggregation methods on the discrete 1-form operator $\mathbb{D}_1^T\mathbb{D}_1$ of a regular hexahedral mesh.

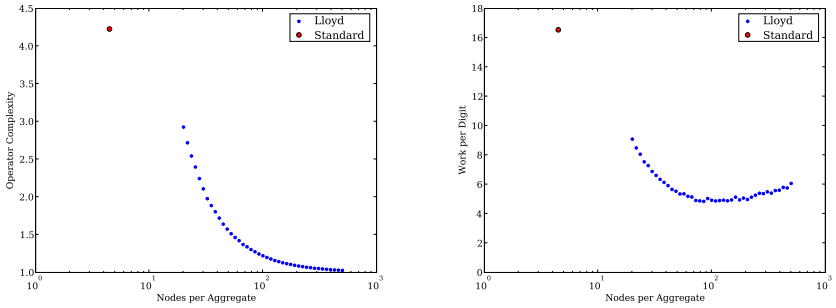

Figure 6.14: Comparison of aggregation methods on the discrete 2-form operator $\mathbb{D}_2^T\mathbb{D}_2$ of a regular hexahedral mesh.

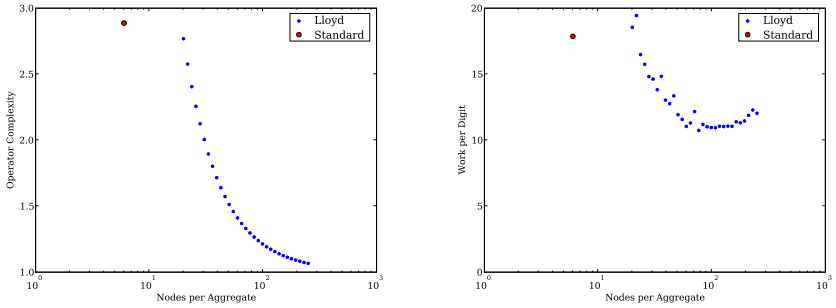

Figure 6.15: Comparison of aggregation methods on the discrete 1-form operator $\mathbb{D}_1^T\mathbb{D}_1$ of an unstructured tetrahedral rocket mesh.

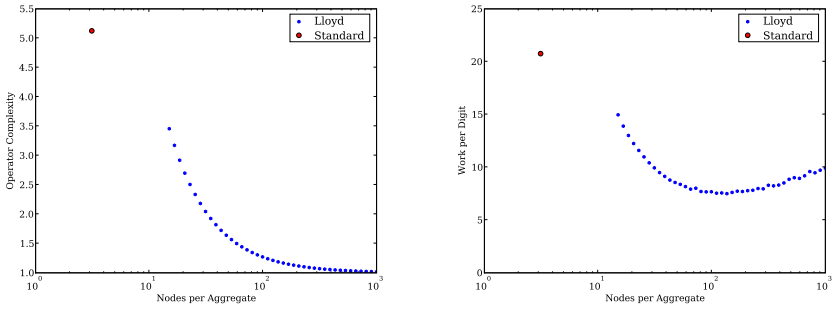

Figure 6.16: Comparison of aggregation methods on the discrete 2-form operator $\mathbb{D}_2^T\mathbb{D}_2$ of an unstructured tetrahedral rocket mesh.

coarsening remains an effective way to reduce setup and memory cost without substantially reducing cycle efficiency.

Chapter 7

Hodge Decomposition

The *Hodge decomposition* [1, 22] states that the space of k-forms on a closed manifold can be decomposed into three orthogonal subspaces,

$$\Omega^k = \boldsymbol{d}_{k-1}\Omega^{k-1} \oplus \boldsymbol{\delta}_{k+1}\Omega^{k+1} \oplus \mathcal{H}^k, \tag{7.1}$$

where $\mathcal{H}^k$ is the space of *harmonic* k-forms, $\mathcal{H}^k = \{h \in \Omega^k | \boldsymbol{\Delta}^k h = 0\}$. Here, $\boldsymbol{\delta}_{k+1}$ denotes the codifferential operator $\boldsymbol{\delta}_{k+1} : \Omega^{k+1} \rightarrow \Omega^k$ which is defined by adjointness to $\boldsymbol{d}_k$,

$$< \boldsymbol{d}_k \alpha^k, \beta^{k+1} > = < \alpha^k, \boldsymbol{\delta}_{k+1} \beta^{k+1} > . \tag{7.2}$$

This chapter introduces the *discrete* Hodge decomposition and its applications. We demonstrate how the multigrid methods discussed in Chapters 4 and 5 are applied to compute Hodge decompositions in the special case $\mathbb{M}_k = I$. A method for computing decompositions in the general case is also presented.

7.1 Discrete Hodge Decomposition

For a discrete k-form ω^k, we seek a decomposition of the form,

$$\omega^k = \mathbb{D}_{k-1}\alpha^{k-1} + \mathbb{M}_k^{-1}\mathbb{D}_k^T\mathbb{M}_{k+1}\beta^{k+1} + h^k, \tag{7.3}$$

for some $\alpha^{k-1} \in \Omega_d^{k-1}$, $\beta^{k+1} \in \Omega_d^{k+1}$, and $h^k \in \Omega_d^k$ where $\boldsymbol{\Delta}^k h^k = 0$. Here, the *derived* codifferential [10]

$$\mathbb{M}_{k-1}^{-1}\mathbb{D}_{k-1}^T\mathbb{M}_k, \tag{7.4}$$

is defined to be the adjoint of $\mathbb{D}_{k-1}$ in the discrete innerproduct $\mathbb{M}_k$. Note that α^{k-1} and β^{k+1} are generally not unique, since the kernels of $\mathbb{D}_{k-1}$ and $\mathbb{M}_k^{-1}\mathbb{D}_k^T\mathbb{M}_{k+1}$ are non-empty. However, the discrete k-forms $(\mathbb{D}_{k-1}\alpha^{k-1})$ and $(\mathbb{M}_k^{-1}\mathbb{D}_k^T\mathbb{M}_{k+1}\beta^{k+1})$ *are* uniquely determined. We decompose ω^k into (7.3) by solving

$$\left(\mathbb{D}_{k-1}^T\mathbb{M}_k\mathbb{D}_{k-1}\right)\alpha^{k-1} = \mathbb{D}_{k-1}^T\mathbb{M}_k\omega^k, \tag{7.5}$$

$$\left(\mathbb{D}_k\mathbb{M}_k^{-1}\mathbb{D}_k^T\right)\mathbb{M}_{k+1}\beta^{k+1} = \mathbb{D}_k\omega^k, \tag{7.6}$$

$$h^k = \omega^k - \mathbb{D}_{k-1}\alpha^{k-1} - \mathbb{M}_k^{-1}\mathbb{D}_k^T\mathbb{M}_{k+1}\beta^{k+1}. \tag{7.7}$$

Note that (7.6) involves the explicit inverse $\mathbb{M}_k^{-1}$ which is typically dense[1]. In the following sections, we first consider the special case $\mathbb{M}_k = I$ and then show how (7.6) can be circumvented in the general case. Equation (7.5) is obtained by left multiplying $\mathbb{M}_{k-1}\mathbb{D}_{k-1}^T\mathbb{M}_k$ on both sides of (7.3). Likewise, applying $\mathbb{D}_k$ to both sides of (7.3) yields (7.6). Equivalently, solutions to (7.5) and (7.6) minimize the functionals

$$Z_1 = \left|\left|\mathbb{D}_{k-1}\alpha^{k-1} - \omega^k\right|\right|_{\mathbb{M}_k}, \tag{7.8}$$

$$Z_2 = \left|\left|\mathbb{M}_k^{-1}\mathbb{D}_k^T\mathbb{M}_{k+1}\beta^{k+1} - \omega^k\right|\right|_{\mathbb{M}_k}, \tag{7.9}$$

respectively. Intuitively, a minimizer of (7.8) is a $(k-1)$-form whose derivative is closest to ω^k in the $\mathbb{M}_k$ norm. Likewise, the discrete codifferential of a minimizer of (7.9) is the $\mathbb{M}_k$-orthogonal projection of ω^k onto the range of $\mathbb{M}_k^{-1}\mathbb{D}_k^T\mathbb{M}_{k+1}$. Convergence of the discrete approximations to the Hodge decomposition is examined in [18].

7.2 Applications

The Hodge decomposition is a fundamental tool in both pure and applied mathematics. For instance, in the context of fluid simulation, the Hodge decomposition provides a means to enforce incompressibility. When representing fluid flow as a discrete 2-form $\omega^2 \in \Omega_d^2$, the divergence of the flow is $\mathbb{D}_2\omega^2$. Therefore, an incompressible, or divergence-free flow satisfies $\mathbb{D}_2\omega^2 = \mathbf{0} \in \Omega_d^3$. Since, in the course of a numerical simulation the fluid flow drifts from the space of divergence-free flows, it is necessary to project the offending component out of the solution. Applying $\mathbb{D}_2$ to the discrete Hodge decomposition (7.3) of ω^2,

$$\mathbb{D}_2\omega^2 = \mathbb{D}_2\mathbb{D}_1\alpha^1 + \mathbb{D}_2\mathbb{M}_2^{-1}\mathbb{D}_2^T\mathbb{M}_3\beta^3 + \mathbb{D}_2h^2, \tag{7.10}$$

$$= \mathbb{D}_2\mathbb{M}_2^{-1}\mathbb{D}_2^T\mathbb{M}_3\beta^3, \tag{7.11}$$

exposes the offending component. Solving for β^3 and then subtracting the 2-form $\mathbb{M}_2^{-1}\mathbb{D}_2^T\mathbb{M}_3\beta^3$ from ω^2 projects the solution back onto the space of incompressible flows. This approach is discussed in the work of Elcott et al. [20].

The visualization of flows [36, 41] is another application of the Hodge decomposition. In complex flows, visualizing each of the three components of the decomposition individually illuminates characteristics that are hidden in a single visualization of the composite.

Furthermore, the Hodge decomposition reveals topological information via differential forms. For example, the two harmonic 1-forms shown in Figure 7.1 exist because the manifold has genus 1. As discussed in Chapter 8, this relationship is utilized in the coordinate-free approach to the sensor network coverage

[1]The covolume Hodge star is a notable exception.

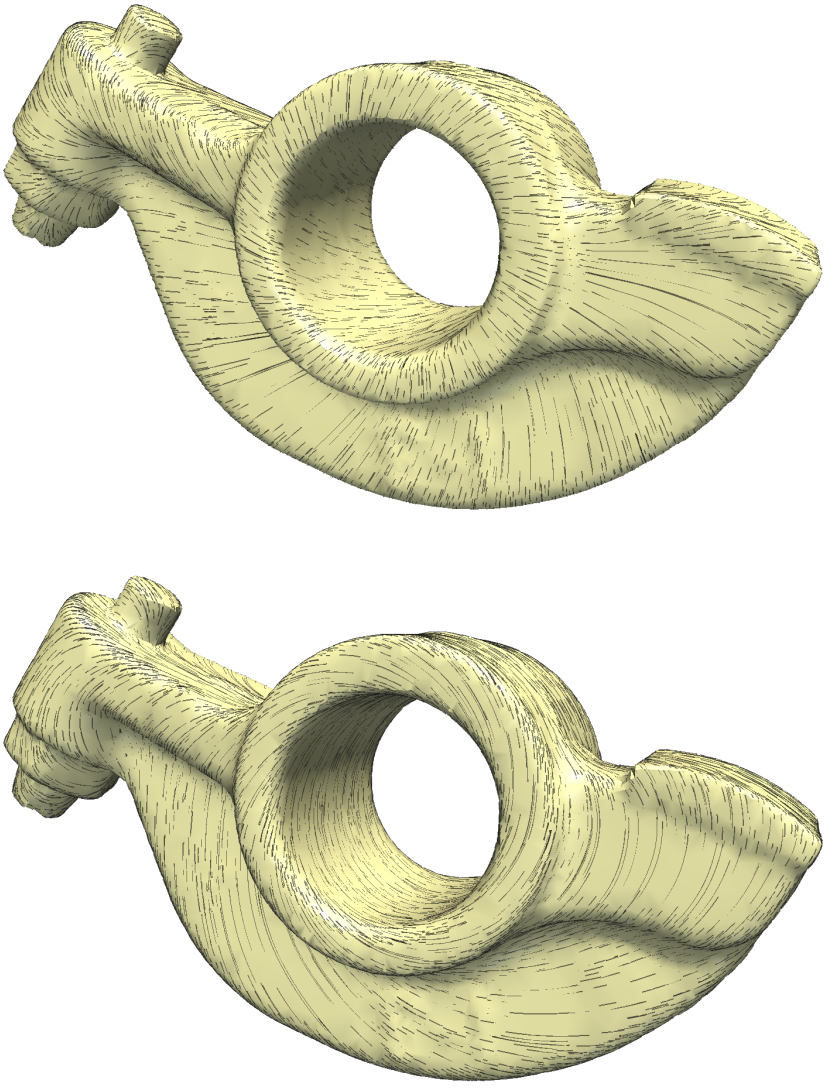

Figure 7.1: The two harmonic 1-forms of a rocker arm surface mesh.

problem.

7.3 Special Case

Taking the appropriate identity matrix for all discrete innerproducts $\mathbb{M}_k$ in (7.5) - (7.7) yields

$$\mathbb{D}_{k-1}^T \mathbb{D}_{k-1} \alpha^{k-1} = \mathbb{D}_{k-1}^T \omega^k, \tag{7.12}$$

$$\mathbb{D}_k \mathbb{D}_k^T \beta^{k+1} = \mathbb{D}_k \omega^k, \tag{7.13}$$

$$h^k = \omega^k - \mathbb{D}_{k-1} \alpha^{k-1} - \mathbb{D}_k^T \beta^{k+1}. \tag{7.14}$$

Equation 7.12 is solved by applying either the cSA method (cf. Chapter 4) or the kSA method (cf. Chapter 5) to the matrix $\mathbb{D}_{k-1}^T \mathbb{D}_{k-1}$. Similarly, either method is applied to (7.13) to compute β^{k+1}.

Although (7.12) - (7.14) are devoid of metric information, some fundamental topological properties of the mesh are retained. For instance, the *number* of harmonic k-forms, which together form a *cohomology basis*, is independent of the choice of innerproduct[2]. In applications where metric information is either irrelevant or simply unavailable[16] these "nonphysical" equations are sufficient.

7.4 General Case

Problems discretized with mimetic finite elements such as Whitney forms [45] give rise to non-trivial discrete innerproducts $\mathbb{M}_k$. Equation 7.5 of the general discrete Hodge decomposition, in which the matrix $\mathbb{D}_{k-1}^T \mathbb{M}_k \mathbb{D}_{k-1}$ appears, is straightforward to solve with either the cSA or kSA methods. However, a different strategy is needed to solve (7.6) since $\mathbb{M}_k^{-1}$ is generally dense and cannot be formed explicitly. In the following, we outline a method for computing Hodge decompositions in the general case.

We first remark that if a basis for the space of Harmonic k-forms, $\mathcal{H}^k = \text{span}\{h_0^k, h_1^k, \ldots h_m^k\}$, is known, then the harmonic component h^k of the Hodge Decomposition is easily computed by projecting ω^k onto the basis elements,

$$h^k = H(H^T \mathbb{M}_k H)^{-1} H^T \mathbb{M}_k \omega^k, \tag{7.15}$$

where $H^k = [h_0^k, h_1^k, \ldots h_m^k]$ denotes the matrix of harmonic k-forms. Furthermore, since (7.5) is solved by cSA or kSA, the value of the remaining component $(\omega^k - \mathbb{D}_{k-1}\alpha^{k-1} - h^k)$, is easily computed. This vector must lie in the range of $\mathbb{M}_k^{-1}\mathbb{D}_k^T \mathbb{M}_{k+1}$ due to orthogonality of the three spaces.

Therefore, the task of computing general Hodge Decompositions is reduced to computing a basis for $\mathcal{H}^k$. Sometimes a basis is known a priori. For instance, $\mathcal{H}^0$ which corresponds to the nullspace of the pure-Neumann scalar problem,

[2] In the case $\mathbb{M} = I$, the cohomology basis is also a *homology basis*.

is spanned by constant vectors on each connected component of the domain. Furthermore, if the domain is *contractible* then $\mathcal{H}^k = \{\}$ for $k > 0$. However, in many cases of interest we cannot assume that a basis for $\mathcal{H}^k$ is known, and therefore it must be computed.

The cSA and kSA solvers are sufficient to determine a Harmonic k-form basis for the identity innerproduct. By decomposing randomly generated k-forms until their respective harmonic components become linearly dependent, a basis, denoted $\{\overline{h_0^k}, \overline{h_1^k}, \ldots \overline{h_m^k}\}$, with span $\overline{\mathcal{H}^k}$ is formed. Using $\{\overline{h_i^k}\}_{i=0}^m$, a basis for the harmonic k-forms with a general innerproduct $\mathbb{M}_k$ is produced by solving

$$\begin{aligned} \mathbb{D}_{k-1}^T \mathbb{M}_k \mathbb{D}_{k-1} \alpha_i^{k-1} &= \mathbb{D}_{k-1}^T \mathbb{M}_k \overline{h_i^k}, & (7.16) \\ h_i^k &= \overline{h_i^k} - \mathbb{D}_{k-1} \alpha_i^{k-1}. & (7.17) \end{aligned}$$

It is readily verified that $h_0^k, \ldots, h_m^k$ are harmonic,

$$\begin{aligned} \mathbb{D}_k h_i^k &= \mathbb{D}_k \overline{h_i^k} - \mathbb{D}_k \mathbb{D}_{k-1} \alpha_i^{k-1}, & (7.18) \\ &= 0 - 0 = 0, & (7.19) \end{aligned}$$

since $\mathbb{D}_k \mathbb{D}_{k-1} = 0$ and $\mathbb{D}_k \overline{h_i^k} = 0$ by assumption, and

$$\begin{aligned} \mathbb{M}_{k-1}^{-1} \mathbb{D}_{k-1}^T \mathbb{M}_k h_i^k &= \mathbb{M}_{k-1}^{-1} (\mathbb{D}_{k-1}^T \mathbb{M}_k \overline{h_i^k} - \mathbb{D}_{k-1}^T \mathbb{M}_k \mathbb{D}_{k-1} \alpha_i^{k-1}), & (7.20) \\ &= \mathbb{M}_{k-1}^{-1}(0) = 0, & (7.21) \end{aligned}$$

by Equation 7.16. It remains to be shown that $h_0^k, \ldots, h_m^k$ are linearly independent.

Suppose otherwise that $h_0^k, \ldots, h_m^k$ are linearly dependent, then there exist scalars $c_0, \ldots, c_H$ not all zero such that

$$\begin{aligned} 0 &= \sum_{i=0}^m c_i h_i^k, \\ &= \sum_{i=0}^m c_i (\overline{h_i^k} - \mathbb{D}_{k-1} \alpha_i^{k-1}), \\ &= \sum_{i=0}^m c_i \overline{h_i^k} - \sum_{i=0}^m c_i \mathbb{D}_{k-1} \alpha_i^{k-1}, \end{aligned}$$

which is a contradiction, since $\left(\sum_{i=0}^m c_i \overline{h_i^k}\right) \in \overline{\mathcal{H}^k}$ is nonzero and $\overline{\mathcal{H}^k} \perp \mathcal{R}(\mathbb{D}_{k-1})$. *Note:* In general, harmonic forms $h_0^k, \ldots, h_m^k$ are not the same as the harmonic components of the random k-forms used to produce $\overline{h_0^k}, \ldots \overline{h_m^k}$.

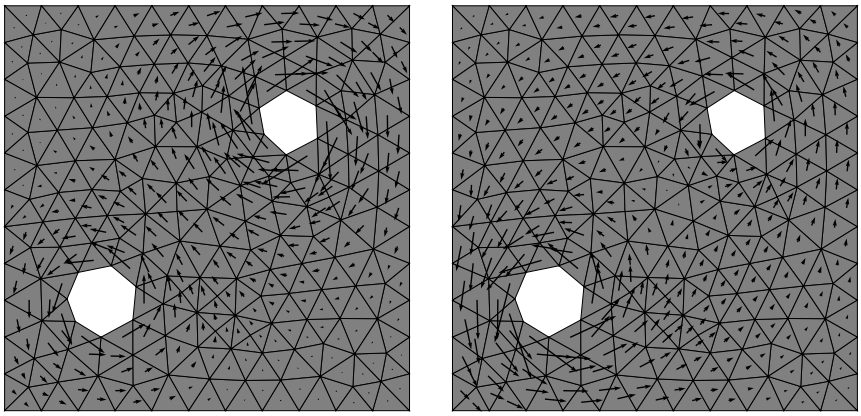

Figure 7.2: Initial harmonic basis.

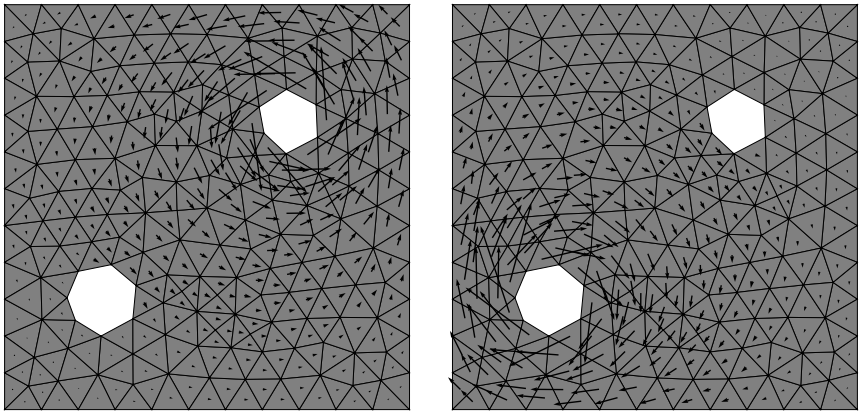

Figure 7.3: Localized harmonic basis.

7.5 Transforming Harmonic Bases

The discrete Hodge decomposition produces a basis for the harmonic k-forms, $H^k = [h_0^k, h_1^k, \ldots h_m^k]$, using the approach discussed in Section 7.4. Although $h_0^k, \ldots h_m^k$ span a basis, the basis vectors do not generally correspond to the most "natural" or "intuitive" representations of the individual features of the manifold. For example, the two harmonic 1-forms illustrated in Figure 7.2 extend across the entire mesh. In contrast, the vectors of an equivalent basis, illustrated in Figure 7.3, localize near the holes of the mesh. In this section we introduce a heuristic that transforms harmonic bases into equivalent, localized bases.

Beginning with an orthonormal set of harmonic basis vectors stored columnwise in the matrix H, Algorithm 7.1 applies Householder transformations [26] to localize the basis. For a given column of the matrix, the heuristic first identifies the element with maximum magnitude and then constructs a House-

Algorithm 7.1: `localize_basis`(H, m)

```
for j = 0 to m
  i ⇐ argmax(abs(H[:,j]))
  v ⇐ H[:,j]
  if H[i,j] > 0:
    v[j] ⇐ v[j] + norm(v)
  else
    v[j] ⇐ v[j] − norm(v)
  end
  Q ⇐ eye(m) − 2 * outer(v,v) / inner(v,v)
  H ⇐ HQ
end
return H
```

holder transformation, denoted Q, to annihilate the corresponding elements in the other columns of H. Intuitively, this procedure pulls spatially overlapping vectors apart at the points where a specific vector takes its maximum value. Figures 7.2 and 7.3 illustrate the basis before and after the localization procedure respectively.

Chapter 8

Sensor Networks

Given a network of *sensors* — e.g. small devices capable of locally monitoring local phenomena or events — a common challenge is to identify holes in the sensor network *coverage*. If the sensors are equipped with a positioning device (e.g. GPS) then the coverage problem reduces to a problem of computational geometry. However, in many cases of interest, the use of positioning hardware may be impractical or too costly. Therefore *coordinate-free* solutions to the coverage problem are desirable.

The coordinate-free approach develops sufficient, but not necessary, conditions for network coverage [16, 33]. We assume that all sensors have fixed *coverage radius* r_c and *broadcast radius*[1] r_b such that $r_c \geq r_b/\sqrt{3}$. Under these assumptions, the difficult geometric problem is replaced with a more-readily computable topological problem: finding a homology basis of the *Rips complex* defined by the network [40]. Furthermore, the coordinate free approach requires only pairwise communication among the sensors. Figure 8.1 illustrates the broadcast radii and Rips complex for a small network of sensors.

In the remainder of this section we discuss the construction of the Rips complex and its associated combinatorial Laplacian operator. We then examine previous numerical methods that have been used to compute homology bases. Finally, we extend our multigrid framework to compute the same homology basis in an efficient and scalable manner.

8.1 Rips Complex

The Rips complex, or Vietoris-Rips complex, is an abstract simplicial complex. Like the de Rham complex (cf. Section 3.2), the Rips complex is equipped with boundary operators $\partial_0, \ldots, \partial_N$ relating simplices to their faces. However, unlike the de Rham complex, which is typically constructed in a top-down fashion from a set of top-level elements (e.g. tetrahedra), the higher-dimensional simplices of the Rips complex are built by recursively combining lower dimensional simplices.

Consider the sensor network illustrated in Figure 8.2 with 300 samples from the unit square. For each pair of points at most r_c distance apart, an edge is included in the complex. Figure 8.3 demonstrates the graph for the case

[1] A sensor communicates with all other sensors within this distance

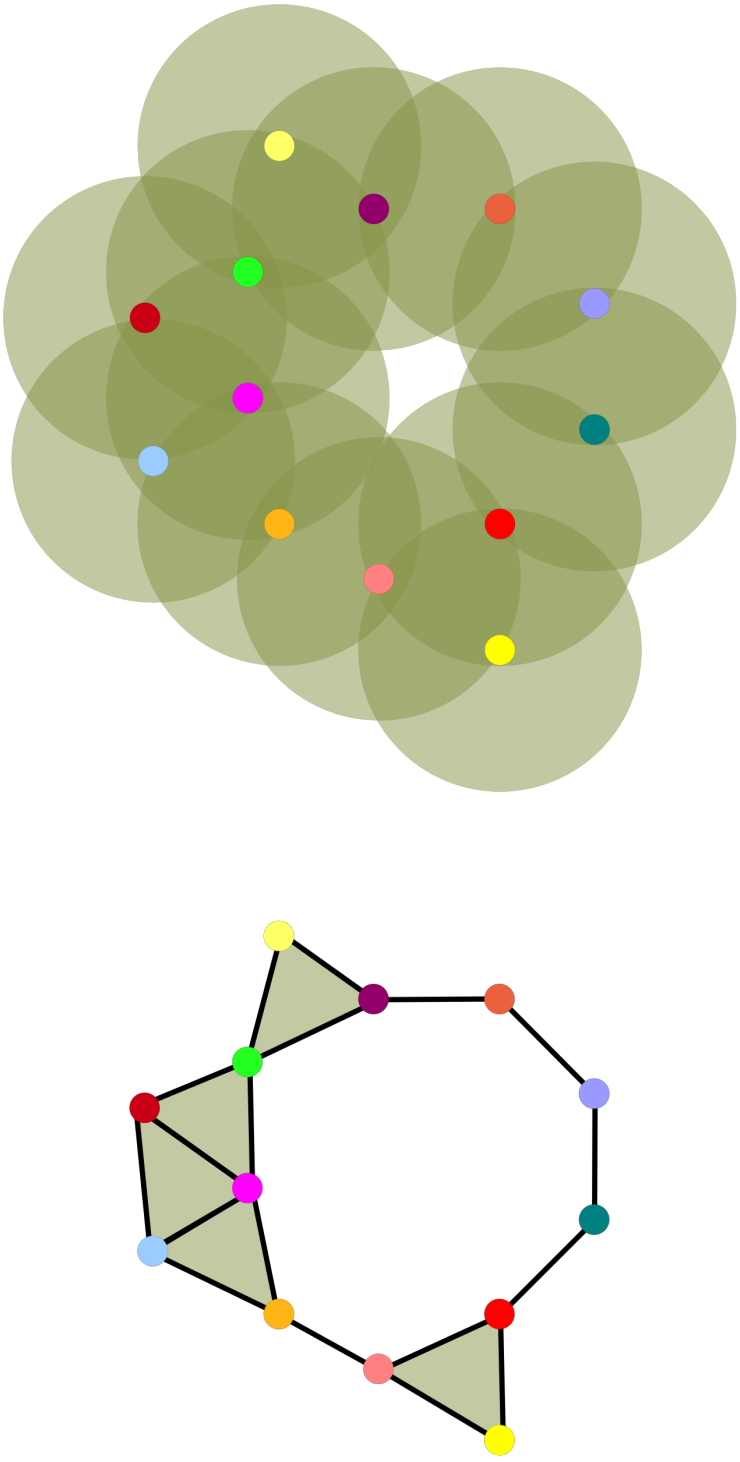

Figure 8.1: Broadcast radii and Rips complex for a sensor network.

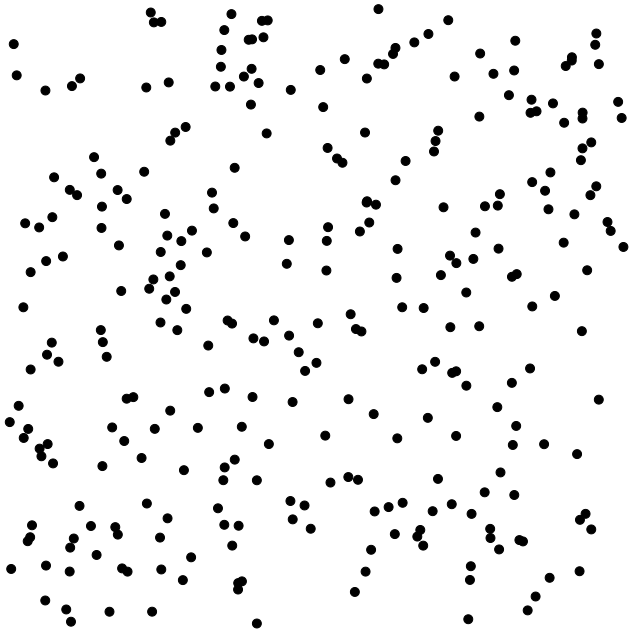

Figure 8.2: Sample sensor network with randomly distributed points.

$r_c = 0.15$. The boundary operator ∂_1 depends on edge orientation, and we (arbitrarily) choose indices in sorted order: (s_0, s_1) where $s_0 < s_1$.

Once the edges of the Rips complex have been chosen, the higher-dimensional simplices are fully determined. A triangle (s_0, s_1, s_2) is added to the Rips complex if and only if all three boundary edges ((s_0, s_1), (s_1, s_2) and (s_0, s_2)) are present in the edge set. In general, a p-simplex is added to the Rips complex when all of its boundary $(p-1)$-faces are present in the complex. Equivalently, each complete graph of $p+1$ nodes in the edge set forms a p-simplex of the Rips complex. In the sensor network problem, it is only necessary to consider p up to the dimension of the embedding space. The Rips complex triangles (2-simplices) for our two dimensional example are shown in Figure 8.4. As with edges, the orientation of higher dimensional p-simplices is chosen arbitrarily. With the boundary operators $\partial_0, \ldots, \partial_N$ determined, we turn our attention to the homology groups of the Rips complex.

8.2 Homology Bases

In order to discover coverage "holes", the coordinate-free approach considers topological properties of the Rips complex which are exposed by the combinatorial Laplacians

$$\Delta_k = \partial_k^T \partial_k + \partial_{k+1} \partial_{k+1}^T \tag{8.1}$$

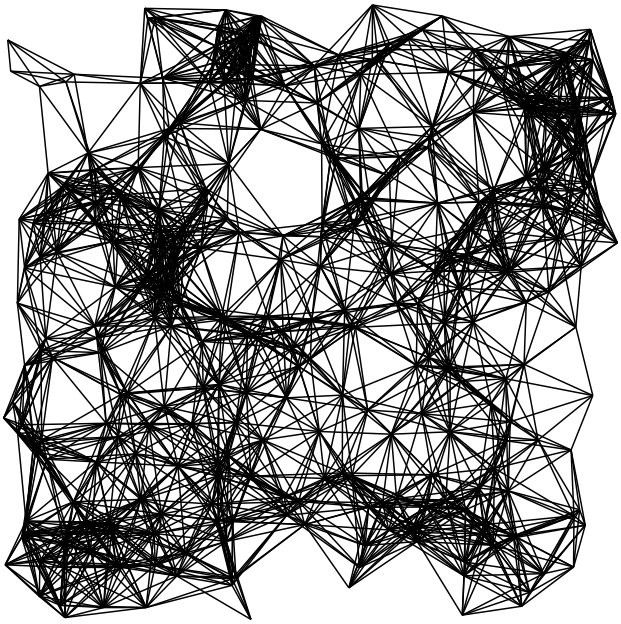

Figure 8.3: Rips complex edge connectivity: Pairs of points within a fixed distance of one another are connected by an edge.

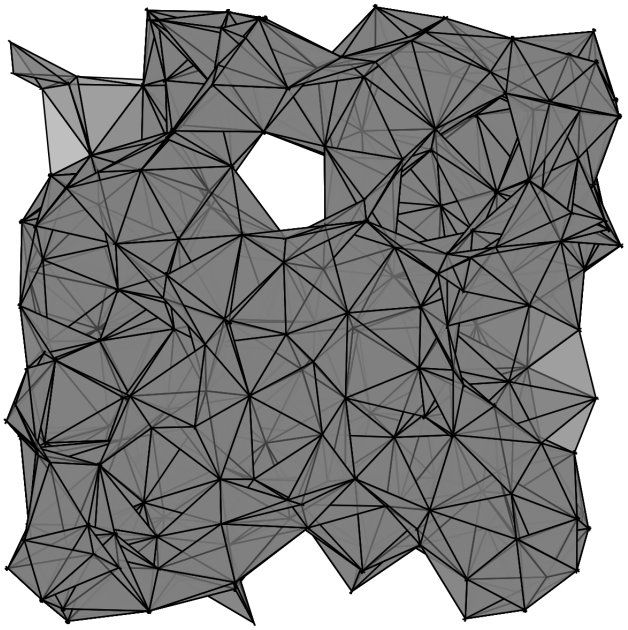

Figure 8.4: Triangles are added to the Rips complex when three points form a clique (complete graph).

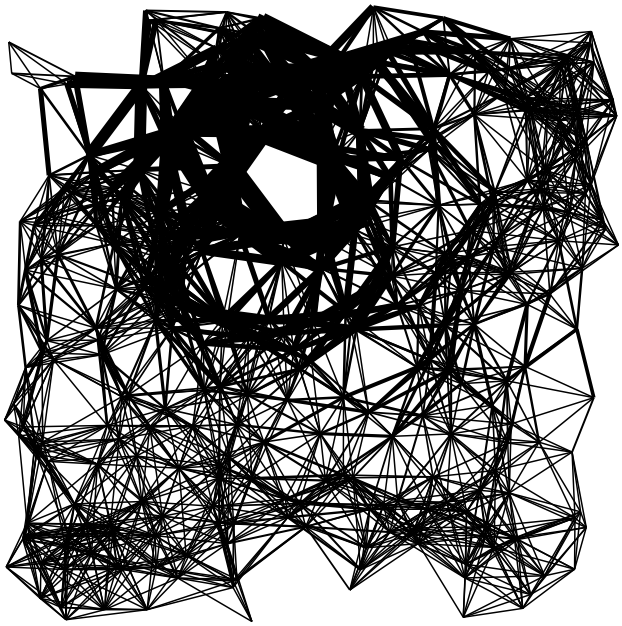

Figure 8.5: The existence of a harmonic 1-form indicates a potential hole in the sensor network coverage. Edge thickness reflects the magnitude of harmonic form on each edge.

derived from the complex [40, 33]. Ultimately, the problem reduces to finding a nullspace basis of the appropriate combinatorial Laplacian for a given spatial dimension. The absence of null-vectors (or harmonic forms) to the combinatorial Laplacian indicates that the Rips complex is free of holes. Conversely, the presense of null-vectors indicates the presence of holes in the Rips complex. However, holes in the Rips complex do not necessarily imply gaps in network coverage.

Consider the (sparse) matrix $\Delta_0 = \partial_0^T \partial_0 + \partial_1 \partial_1^T$. Since ∂_0 is the zero matrix, $\Delta_0 = \partial_1 \partial_1^T$ is precisely the *graph Laplacian* of the Rips complex. A well-known property of the graph Laplacian is that its nullspace basis has precisely one vector for each connected component of the graph. Therefore, the (somewhat trivial) problem of counting graph components reduces to finding a null-basis of Δ_0.

In the context of sensor networks, the combinatorial Laplacian of interest is Δ_1 in two dimensions and Δ_2 in three dimensions. Continuing our example, we consider the nullspace of Δ_1 for the Rips complex shown in Figure 8.4. Since the visual representation of the complex has one hole we expect to find a single harmonic form. This vector, which has a scalar value for each edge of the complex, is shown in Figure 8.5. In general, a two dimensional complex with K holes has K harmonic 1-forms. Similarly, a three dimensional complex with K holes, or voids, has K harmonic 2-forms, or nullvectors of Δ_2.

8.3 Numerical Methods

We now consider numerical methods finding nullspace bases of Δ_K. Muhammad and Egerstedt [33] pose the problem as a dynamical system with steady states corresponding to harmonic forms. Specifically, a random vector ω^k is evolved in fictitious time by the equation

$$\frac{\partial \omega}{\partial t} = -\Delta_k \omega^k. \tag{8.2}$$

Numerical integration of a forward-Euler time discretization of Equation 8.2 is equivalent to applying Richardson iteration to the linear system

$$\Delta_k \omega^k = 0, \tag{8.3}$$

with the same initial vector. Although this approach is simple and naturally parallelizes over the sensor network, the slow rate of convergence results in a large number of iterations.

In [23], the Power Method is applied to produce null-vectors of the combinatorial Laplacian. Like the dynamical system approach, this method also corresponds to Richardson iteration on (8.3) and therefore exhibits the same convergence.

Standard iterative methods with better convergence behavior than Richardson iteration exist. For instance, conjugate gradient iteration (CG) or the MINRES method [35] on the positive semi-definite system (8.3) converges to the nullspace component of the initial starting vector. Although these methods converge to an approximate solution in fewer iterations than Richardson's method, each iteration requires a vector innerproduct, which in the context of sensor networks, necessitates a global reduction operation. However, since the cost of a global reduction is significant, perhaps proportional to the diameter of the network, methods that converge in fewer iterations are not necessarily faster.

8.4 Proposed Method

Like the approaches described in Section 8.3, we iteratively solve Equation 8.3 with a random initial vector to produce a nullvector of Δ_k. We use the MINRES method [35] with an adaptive kSA preconditioner (cf. Section 5.6). As a black-box solver, the adaptive kSA method requires no coordinate information. Although both the preconditioner and the outer MINRES iteration require global reductions at each iteration, the number of required iterations is small.

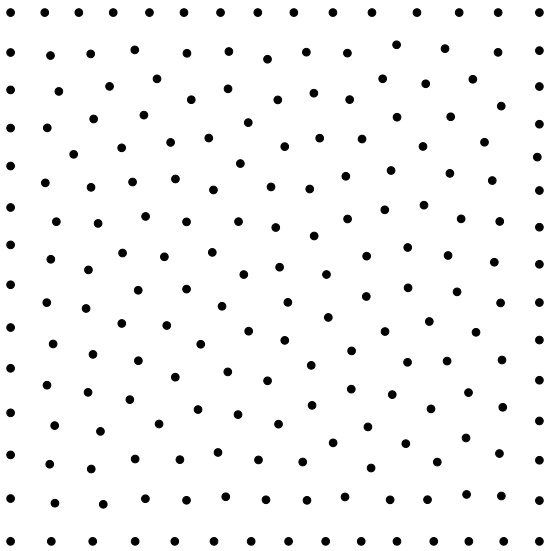

Figure 8.6: Distribution of 200 points produced by repulsion method.

8.5 Numerical Results

Since no established manner of simulating the distribution of points in real-world sensor networks exists, we propose a straightforward ad hoc approach. Beginning with a set of N points uniformly distributed from the unit square (or cube), a repulsive force pushes nearby points apart. The i-th point, with coordinate vector v_i, is integrated through fictitious time according to the ODE

$$\frac{\partial v_i}{\partial t} = \sum_{j \in B_R(v_i), i \neq j} \left(1 - \frac{||v_i - v_j||}{R}\right)^2 \frac{v_i - v_j}{||v_i - v_j||}, \tag{8.4}$$

where $R > 0$ denotes a given radial distance, and $B_R(v_i)$ the ball of radius R about the point v_i. Intuitively, Equation 8.4 repels all points within distance R, with a magnitude that approaches zero as the separation approaches R. In our implementation, R is $2\sqrt[D]{1.0/N}$, where D is the dimension of the space, and 30 steps of Forward Euler integration are used to advance Equation 8.4 through time. As shown by Figure 8.6, the resulting points are evenly distributed.

In order to test the scalability of the proposed method, we use the repulsion method to produce distributions with between 100 and 100,000 points in both two and three dimensions. In each case, we compute the Rips complex for the broadcast radius $r_b = 2\sqrt[D]{1.0/N}$, the same distance used in the repulsion model. Figure 8.7 illustrates the Rips complex for the point distribution shown in Figure 8.6 with this choice of r_b.

In this section we report performance figures for the proposed method applied to matrices Δ_1 and Δ_2. Unless stated explicitly, we apply the same testing methodology used in Section 5.6. For the purposes of comparison we retain the

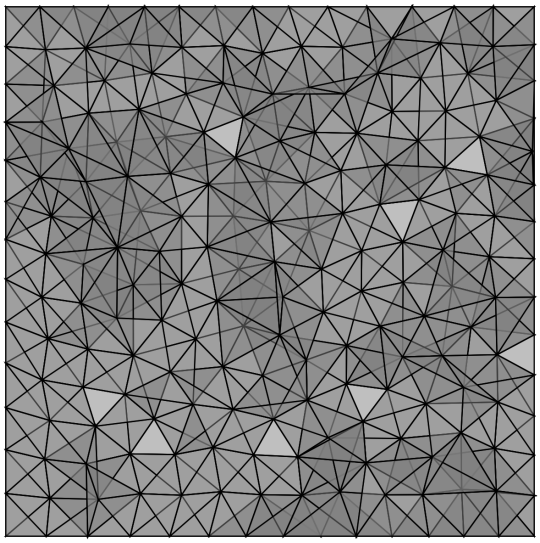

Figure 8.7: Rips complex for 200 points produced by the repulsion method.

System	Pts	Unknowns	Convergence	WPD	OC	Lvls
Δ_1	100	377	0.220	6.369	1.046	2
	200	810	0.289	8.355	1.125	2
	500	2,366	0.299	9.151	1.198	2
	1,000	4,858	0.349	10.862	1.239	2
	2,000	10,701	0.321	10.551	1.301	2
	5,000	27,786	0.326	11.150	1.357	3
	10,000	56,291	0.362	12.546	1.383	3
	20,000	114,305	0.397	14.073	1.408	3
	50,000	288,857	0.419	15.085	1.421	4
	100,000	580,157	0.475	17.608	1.421	4

Table 8.1: Solver performance on hole-free sensor networks in 2D.

use of symmetric Gauss-Seidel for pre- and post-smoothing. In practice, a parallel smoother [3] is more appropriate for this application.

Table 8.1 reports numerical results for the proposed method on a series of two-dimensional, hole-free sensor networks. In each case, six candidates were used during the adaptive setup phase. Each first-level aggregate, computed with Lloyd aggregation, consists of an average of 150 unknowns. The 'Pts' column of Table 8.1 refers to the number of sensors in the network while 'Unknowns' refers to the number of edges in the complex. As the number of points is increased there is a modest rise in the rate of convergence and overall work per digit of accuracy. While convergence of the method is not independent of problem size, it does not substantially deteriorate.

The three dimensional results in Table 8.2 exhibit similar dependence on problem size. However, as in the previous case, the degradation in performance

System	Points	Unknowns	Convergence	WPD	OC	Lvls
Δ_1	100	534	0.176	5.344	1.005	2
	200	1,323	0.248	6.623	1.002	2
	500	4,113	0.296	7.641	1.007	2
	1,000	9,721	0.335	8.543	1.012	2
	2,000	21,630	0.354	9.103	1.025	2
	5,000	60,525	0.385	10.085	1.043	2
	10,000	129,371	0.376	10.017	1.063	3
	20,000	272,599	0.388	10.428	1.070	3
	50,000	717,259	0.441	12.232	1.086	3
	100,000	1,481,416	0.430	11.969	1.094	3
Δ_2	100	890	0.221	6.140	1.005	2
	200	2,697	0.303	7.732	1.001	2
	500	10,280	0.373	9.400	1.005	2
	1,000	28,494	0.408	10.391	1.009	2
	2,000	69,621	0.440	11.409	1.015	2
	5,000	215,366	0.466	12.427	1.027	2
	10,000	488,461	0.483	13.122	1.035	3
	20,000	1,077,654	0.496	13.724	1.042	3
	50,000	2,961,027	0.517	14.645	1.047	3
	100,000	6,292,204	0.541	15.767	1.051	3

Table 8.2: Solver performance on hole-free sensor networks in 3D.

is subtle. Since the number of unknowns per point is larger in three dimensions, more rapid coarsening has been used. First level aggregates consist of (on average) 500 and 2500 unknowns in the 1-form and 2-form problems respectively. Six candidates are used in the adaptive setup phase in all tests. Although the 1-form problem Δ_1 is not needed to determine sensor coverage in 3D, results are included for completeness.

Although the Rips complex is a collection of simplices which connect points in close proximity, the structure of the Rips complex is unlike that of a standard finite-element mesh. For instance, the tetrahedral rocket mesh displayed in Figure 4.4 has an average of 5.3 tetrahedra per vertex while the three-dimensional Rips complex with 100,000 points has 121.4 tetrahedra per vertex. Redundant overlaps, such as those in Figure 8.4, are responsible for the large number of higher-dimensional simplices.

While, in the cases presented, the proposed method is sensitive to problem size, it represents a profound improvement over the previously applied numerical methods discussed in Section 8.3. Figure 8.8 compares the convergence behavior of the proposed method to other iterative solvers on the two-dimensional network with 100,000 points. Richardson iteration, which is equivalent to the approaches of [33] and [23], converges very slowly. Though, on a per-iteration basis, (non-preconditioned) conjugate gradient iteration improves on Richardson's method, both solvers remain substantially slower than the proposed multigrid method.

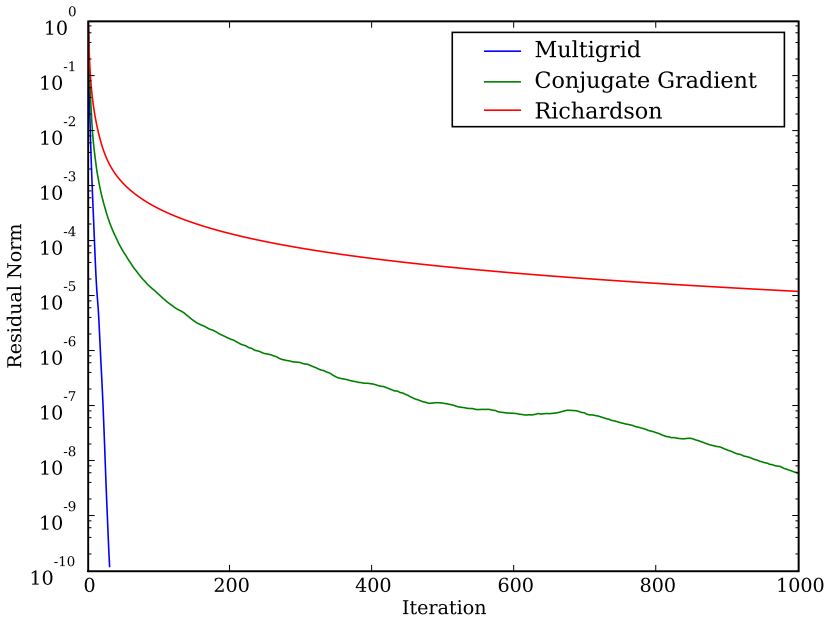

Figure 8.8: Convergence history of several iterative methods applied to a hole-free two-dimensional sensor network with 100,000 points in 2D. The proposed multigrid method reduces the residual norm $||b-Ax||$ by 10 orders of magnitude in 31 iterations. Richardson iteration and the conjugate gradient method fail to reach the same threshold within 1000 iterations.

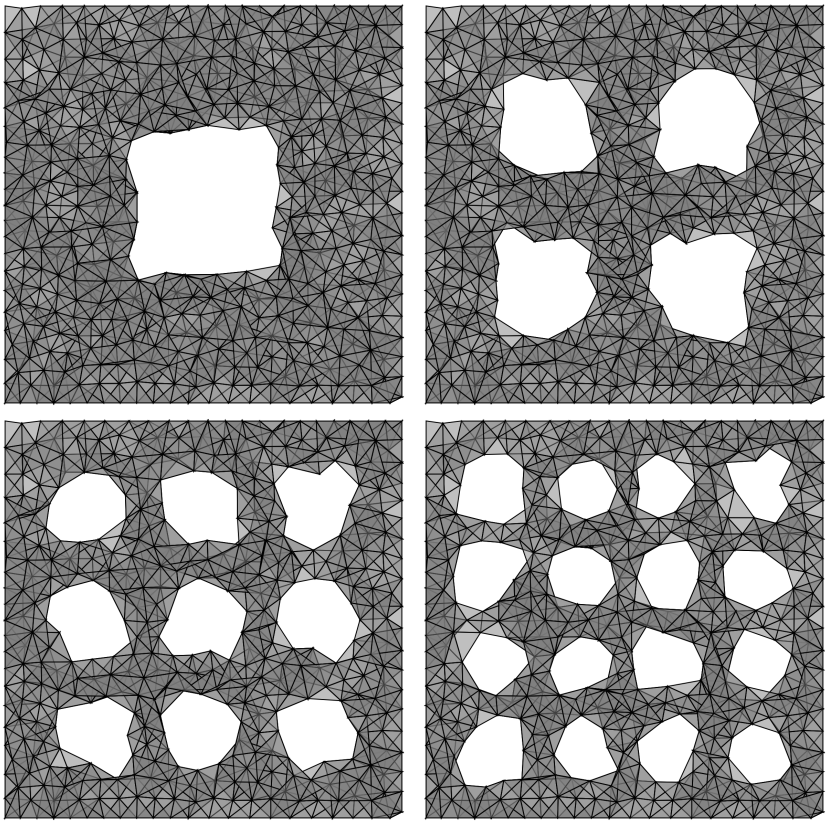

Figure 8.9: Sensor networks with an increasing number of coverage holes.

We now consider solver performance on sensor networks with one or more coverage holes. The following data sets are constructed from the two and three-dimensional sensor networks with 10,000 points by removing points within certain regions of the domain. Figure 8.9 illustrates the patterns used in two dimensions. A similar pattern is applied in the three-dimensional case.

Table 8.3 reports the performance of the proposed method in two dimensions with 1, 2, 4, 9 and 16 holes. For comparison, figures for the hole-free sensor network (cf. Table 8.1) are included. As the number of holes is increased, the convergence ratio and work per digit of accuracy remain steady. Therefore, in the tests considered, the proposed method is insensitive to the presence of holes. Figure 8.10 illustrates several candidate vectors produced by the adaptive setup phase on a Rips complex with four holes. The candidates are consistent with of our expectations regarding the nullspace of Δ_1.

Performance results of the three-dimensional sensor networks with 1, 8, 27, and 64 holes, or voids, are listed in Table 8.4. The presence of holes yields a modest degradation in convergence and work per digit of accuracy in the Δ_1 tests. On the other hand, solver performance on the Δ_2 system is insensitive to the topology of the complex. Again, only the 2-form problem is needed in the

System	Holes	Pts	Unknowns	Convergence	WPD	OC	Lvls
Δ_1	0^2	10,000	56,291	0.362	12.546	1.383	3
	1^2	8,888	49,370	0.310	12.177	1.547	3
	2^2	8,358	45,467	0.310	11.872	1.506	3
	3^2	8,167	44,219	0.329	12.215	1.471	3
	4^2	8,043	43,114	0.298	11.009	1.446	3

Table 8.3: Solver performance on 2D sensor networks in with holes.

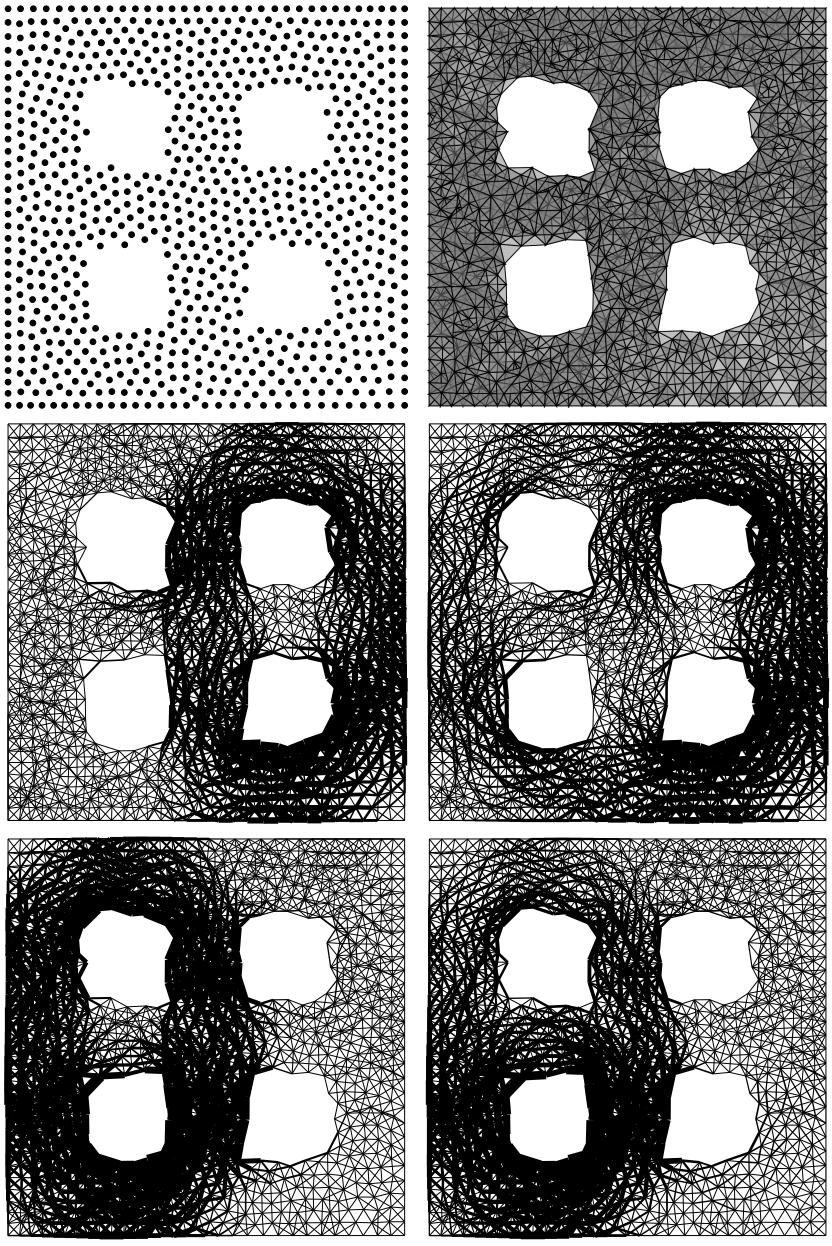

Figure 8.10: Candidate vectors computed during the adaptive setup phase.

System	Holes	Pts	Unknowns	Convergence	WPD	OC	Lvls
Δ_1	0^3	10,000	129,371	0.376	10.017	1.063	3
	1^3	9,664	122,784	0.340	9.004	1.054	3
	2^3	9,117	111,555	0.353	9.306	1.051	3
	3^3	8,478	112,918	0.449	12.120	1.053	3
	4^3	7,924	97,033	0.554	16.358	1.048	3
Δ_2	0^3	10,000	488,461	0.483	13.122	1.035	3
	1^3	9,664	457,670	0.425	11.118	1.031	3
	2^3	9,117	430,801	0.445	11.725	1.030	3
	3^3	8,478	360,304	0.376	9.666	1.025	3
	4^3	7,924	298,377	0.475	12.697	1.024	3

Table 8.4: Solver performance on 3D sensor networks with holes.

context of three-dimensional coverage problems.

Chapter 9

Conclusions

This thesis develops efficient and scalable numerical solvers for discrete k-form problems. We summarize the main contributions of the thesis in this chapter.

9.1 Contributions

Development and analysis of cSA. We have described an extension of Reitzinger and Schöberl's methodology [37] to higher dimensional k-forms with the addition of smoothed prolongation operators. Furthermore, we have detailed properties of the prolongation operators that arise from this generalized setting. Specifically we have identified necessary and sufficient conditions under which commutativity is maintained. The prolongation operators give rise to a hierarchy of compatible finite element spaces. The generality of the method is appealing since the components are constructed independently of a particular mimetic discretization. We have demonstrated problem-size-independent convergence of the cSA method in the context of structured meshes.

Development and analysis of kSA. We have introduced the k-form basis method (kSA) for solving discrete k-form problems. The kSA method improves on cSA by ensuring that the tentative prolongation operators used in the SA setup procedure reproduce a set of k-form basis functions. As a result, solver performance for problems on unstructured meshes does not degrade when a non-trivial innerproduct $\mathbb{M}_k$ is introduced. On manifolds with multiple parameterizations, we have explored the use of different k-form bases, those that are intrinsic to the manifold and those given by an embedding. Lastly, we have considered the case of combinatorial Laplacians and an adaptive method based on kSA with which to solve them.

Development and analysis of Lloyd Aggregation. We have demonstrated that Lloyd aggregation is an effective aggregation algorithm for a variety of problem discretizations. In the context of dual-mesh and k-form problems, the standard aggregation algorithm [44], which is tailored for matrices arising in standard (primal) nodal discretizations, is significantly more costly than Lloyd aggregation. On such problems, Lloyd aggregation-based solvers require less work per digit of accuracy while maintaining

lower operator complexity than the standard method. Lloyd aggregation supports a time-memory tradeoff between work per digit of accuracy (WPD) and operator complexity. Although WPD is generally the primary measure of multigrid performance, the memory constraints of a given platform place a hard constraint on the allowable operator complexity. In this situation, the ability of Lloyd aggregation to produce hierarchies with arbitrarily low operator complexity is advantageous. Furthermore, the use of Lloyd aggregation with an aggressive coarsening rate does not lead to an immediate increase in work per digit of accuracy. Instead, we have shown that there is a gradual tradeoff between operator complexity and WPD over a wide range of coarsening rates.

Efficient computation of discrete Hodge decompositions. The discrete Hodge decomposition is an important tool with numerous applications in the computational sciences. We have initiated a study of algebraic multigrid for the Hodge decomposition of discrete k-forms. The direct application of the cSA and kSA methods is sufficient to compute decompositions in the special case $\mathbb{M}_k = I$. Furthermore, we have described a method to compute decompositions for general innerproducts without constructing the explicit inverse of $\mathbb{M}_k$.

Efficient determination of sensor network coverage. We have applied the adaptive kSA methodology to the sensor network coverage problem. Like previous approaches, the proposed method computes nullvectors of the combinatorial Laplacian Δ_k by solving $\Delta_k \omega^k = 0$ with an iterative solver. However, our results demonstrate that the adaptive kSA solver converges to an approximate solution far more rapidly than those of previous approaches (e.g. Richardson iteration). While the convergence of the proposed method is not independent of network size, the performance degradation is mild, and does not preclude use of the solver in large-scale problems.

9.2 Closing Remarks

The numerical experiments and results of this thesis were conducted with PyAMG [7] a collection of algebraic multigrid solvers with a Python interface. PyAMG implements several AMG methods such as classical AMG [38], smoothed aggregation (SA) [44], and adaptive smoothed aggregation [14]. Since the components of these methods are modular, PyAMG provides an ideal environment for rapidly prototyping new multigrid methods. Furthermore, since costly operations are implemented with natively-compiled programming languages, large-scale problems are solved efficiently.

References

[1] R. Abraham, J. E. Marsden, and T. Ratiu. *Manifolds, Tensor Analysis, and Applications*. Springer–Verlag, New York, second edition, 1988.

[2] M. Adams. A parallel maximal independent set algorithm. *Proceedings 5th copper mountain conference on iterative methods*, 1998.

[3] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *J. Comput. Phys.*, 188(2):593–610, 2003.

[4] D. N. Arnold. Differential complexes and numerical stability. In *Proceedings of the International Congress of Mathematicians, Beijing 2002, Volume 1 : Plenary Lectures*, 2002.

[5] D. N. Arnold, R. S. Falk, and R. Winther. Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$. *Numer. Math.*, 85(2):197–217, 2000.

[6] N. Bell and L. Olson. Algebraic multigrid for k-form laplacians. *Numerical Linear Algebra with Applications*, 15(2–3):165–185, 2008.

[7] N. Bell, L. Olson, and J. Schroder. PyAMG : Algebraic multigrid solvers in Python. `http://www.pyamg.com/`, 2007-.

[8] P. Bochev, J. Hu, C. Siefert, and R. Tuminaro. An Algebraic Multigrid Approach Based on a Compatible Gauge Reformulation of Maxwells Equations. 2007.

[9] P. Bochev, C. Siefert, J. Hu, and R. Tuminaro. An algebraic multigrid approach based on a compatible gauge reformulation of Maxwell's equations. 2007.

[10] P. B. Bochev and J. M. Hyman. Principles of mimetic discretizations of differential operators. In D. N. Arnold, P. B. Bochev, R. B. Lehoucq, R. A. Nicolaides, and M. Shashkov, editors, *Compatible Spatial Discretizations*, volume 142 of *The IMA Volumes in Mathematics and its Applications*, pages 89–119. Springer, Berlin, 2006.

[11] P. B. Bochev and A. C. Robinson. Matching algorithms with physics: exact sequences of finite element spaces. In D. Estep and S. Tavener, editors, *Collected Lectures on Preservation of Stability Under Discretization*, chapter 8, pages 145–166. Society for Industrial and Applied Mathematics (SIAM), 2002.

[12] A. Bossavit. On the numerical analysis of eddy-current problems. *Computer Methods in Applied Mechanics and Engineering*, 27(3):303–318, 1981.

[13] A. Bossavit. Whitney forms : a class of finite elements for three-dimensional computations in electromagnetism. *IEE Proceedings*, 135, Part A(8):493–500, November 1988.

[14] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. Adaptive smoothed aggregation (αsa) multigrid. *SIAM Rev.*, 47(2):317–346, 2005.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[16] V. de Silva and R. Ghrist. Homological Sensor Networks. *Notices of the American Mathematical Society*, 54:10–17, 2007.

[17] M. Desbrun, E. Kanso, and Y. Tong. Discrete differential forms for computational modeling. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 39–54, New York, NY, USA, 2006. ACM.

[18] J. Dodziuk. Finite-difference approach to the Hodge theory of harmonic forms. *Amer. J. Math.*, 98(1):79–104, 1976.

[19] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[20] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun. Stable, circulation-preserving, simplicial fluids. Unpublished, 2005.

[21] M. Fisher, P. Schröder, M. Desbrun, and H. Hoppe. Design of tangent vector fields. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 56, New York, NY, USA, 2007. ACM.

[22] T. Frankel. *The Geometry of Physics*. Cambridge University Press, Cambridge, second edition, 2004. An introduction.

[23] J. Friedman. Computing Betti Numbers via Combinatorial Laplacians. *Algorithmica*, 21(4):331–346, 1998.

[24] V. Gradinaru and R. Hiptmair. Whitney Elements on Pyramids. *Electronic Transactions on Numerical Analysis*, 8:154–168, 1999.

[25] X. Gu and S.-T. Yau. Global conformal surface parameterization. In L. Kobbelt, P. Schröder, and H. Hoppe, editors, *Eurographics Symposium on Geometry Processing*. Eurographics, 2003.

[26] M. T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, second edition, 2002.

[27] R. Hiptmair. Multigrid method for maxwell's equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1999.

[28] A. N. Hirani. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology, May 2003.

[29] J. J. Hu, R. S. Tuminaro, P. B. Bochev, C. J. Garasi, and A. C. Robinson. Toward an h-independent algebraic multigrid method for Maxwell's equations. *SIAM Journal on Scientific Computing*, 27:1669–1688, 2006.

[30] S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[31] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1055, 1986.

[32] J. Mandel, M. Brezina, and P. Vaněk. Energy optimization of algebraic multigrid bases. *Computing*, 62(3):205–228, 1999.

[33] A. Muhammad and M. Egerstedt. Control Using Higher Order Laplacians in Network Topologies. *Proc. of 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan*, pages 1024–1038, 2006.

[34] R. A. Nicolaides and K. A. Trapp. Covolume discretization of differential forms. In D. N. Arnold, P. B. Bochev, R. B. Lehoucq, R. A. Nicolaides, and M. Shashkov, editors, *Compatible Spatial Discretizations*, volume 142 of *The IMA Volumes in Mathematics and its Applications*. Springer, Berlin, 2006.

[35] C. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[36] K. Polthier and E. Preuss. Identifying vector field singularities using a discrete hodge decomposition. In H. C. Hege and K. Polthier, editors, *Visualization and Mathematics, VisMath*. Springer–Verlag, 2002.

[37] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numer. Linear Algebra Appl.*, 9:223–238, 2002.

[38] J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, volume 3 of *Frontiers Appl. Math.*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[39] J. Schroder. A General Strength-of-Connection Concept in AMG. *Tenth Copper Mountain Conference on Iterative Methods*, 2008.

[40] V. D. Silva and R. Ghrist. Coordinate-free coverage in sensor networks with controlled boundaries via homology. *Int. J. Rob. Res.*, 25(12):1205–1222, 2006.

[41] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. *ACM Transactions on Graphics (Special issue of SIGGRAPH 2003 Proceedings)*, 22(3):445–452, July 2003.

[42] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 1 edition, 2000.

[43] E. Vanderzee, A. N. Hirani, E. Ramos, and D. Guoy. Well-centered meshing. In preparation, 2006.

[44] P. Vaněk, J. Mandel, and M. Brezina. Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems. *Computing*, 56(3):179–196, 1996.

[45] H. Whitney. *Geometric Integration Theory*. Princeton University Press, Princeton, N. J., 1957.

[46] K. S. Yee. Numerical solution of initial boundary value problems involving Maxwells equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, AP-14(3):302–307, May 1966.

Curriculum Vitae

Research Interests

Multigrid methods for discrete differential forms, discrete Hodge decompositions, computational homology/cohomology, and iterative methods for linear systems.

Education

University of Illinois at Urbana-Champaign, Urbana, Illinois

Ph.D. Computer Science, August 2008

Georgia Institute of Technology, Atlanta, Georgia

B.S. Computer Science, August 2003

B.S. Discrete Mathematics, August 2003

Publications

Algebraic Multigrid for k-form Laplacians

Nathan Bell and Luke Olson

Numerical Linear Algebra with Applications, 15:2-3, 165-185, 2008.

Particle-Based Simulation of Granular Materials

Nathan Bell, Yizhou Yu, and Peter J. Mucha

ACM SIGGRAPH Symposium on Computer Animation 2005

A Fast Multigrid Algorithm for Mesh Deformation

Lin Shi, Yizhou Yu, Nathan Bell and Wei-Wen Feng

ACM Transactions on Graphics, Proceedings of SIGGRAPH 2006

Honors and Awards

Outstanding Teaching Assistant

Department of Computer Science

University of Illinois at Urbana-Champaign, Spring 2006

Best Paper

ACM SIGGRAPH Symposium on Computer Animation 2005

Incomplete List of Teachers Ranked as Excellent

University of Illinois at Urbana-Champaign, Spring 2004

Employment

NVIDIA Corporation, Santa-Clara, California

Research Intern

University of Illinois at Urbana-Champaign, Urbana, Illinois

Teaching Assistant

Landauer Incorporated, Glenwood, IL

Research and Development Intern

Software Contributions

PyAMG

Algebraic Multigrid Solvers in Python

http://www.pyamg.com/

SciPy

Open-source software for mathematics, science, and engineering

http://www.scipy.org/